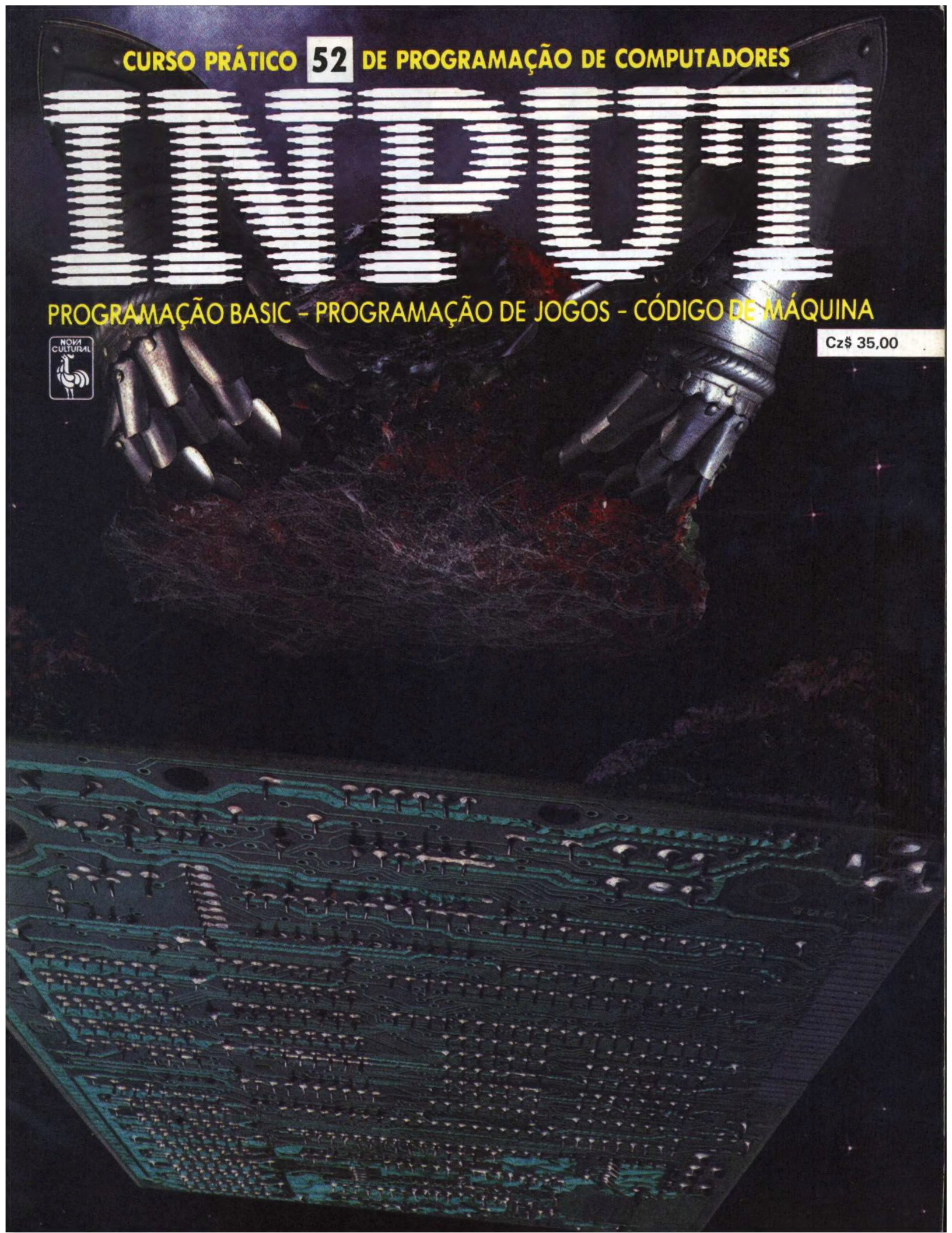
CURSO PRÁTICO 52 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 35,00



INPUT

Vol. 4 N.º 52

NESTE NÚMERO

PROGRAMAÇÃO BASIC

MONTAGEM DE DESENHOS

Princípios básicos do PAC. Movimentos na tela. Expansão, contração e rota de figuras ... 1021

CÓDIGO DE MÁQUINA

EFEITOS SONOROS COMPLEXOS

Como obter o efeito de um "tiro laser". Movimentos do cone e pausas. Duração 1027

CÓDIGO DE MÁQUINA

AVALANCHE: O VÔO DAS GAIVOTAS

Variável de atraso. Como fazer a gaivota bater asas. Decolagem. Definição do vôo..... 1028

CÓDIGO DE MÁQUINA

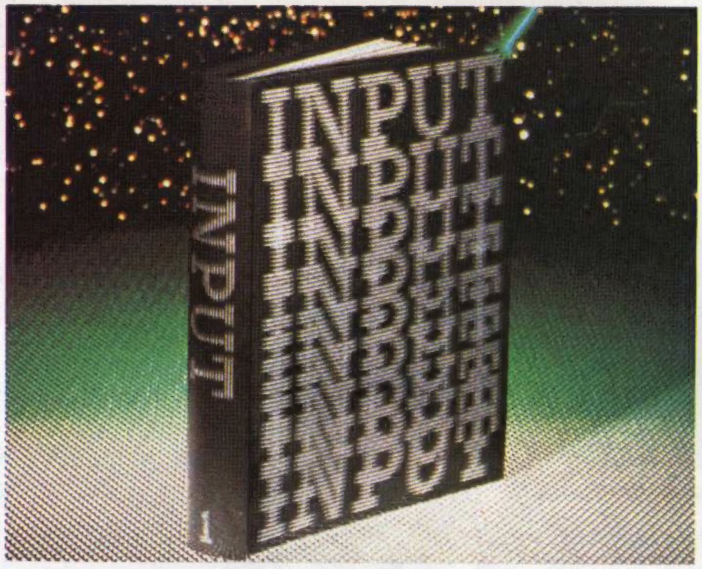
SONS E RUÍDOS NO TRS-80

A porta de saída. Tons musicais. Como variar a freqüência. Tiros e explosões 1032

PROGRAMAÇÃO DE JOGOS

JOGOS DE GUERRA: O MAPA DA BATALHA

Matrizes do mapa e das tropas. Colocação dos blocos gráficos na tela. Moldura 1034



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/ Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

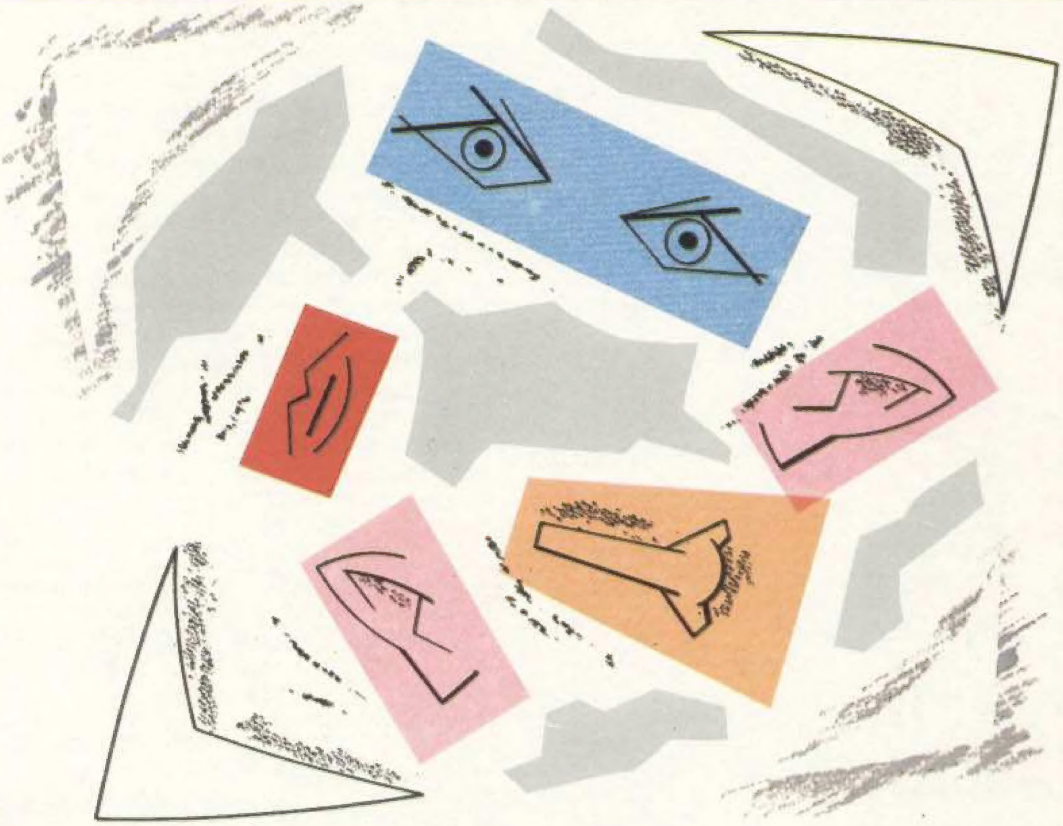
© Marshall Cavendish Limited 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

MONTAGEM DE DESENHOS

- PRINCÍPIOS BÁSICOS DO PAC
- LIGANDO PONTOS
- MOVIMENTAÇÃO DA FIGURA
- EXPANSÃO, CONTRAÇÃO E ROTAÇÃO

A construção de imagens gráficas no computador é, quase sempre, um processo difícil. O Projeto Assistido pelo Computador (PAC) poderá simplificar bastante o seu trabalho.



Existem sistemas destinados a facilitar a elaboração de desenhos cujos componentes se repetem muitas vezes. O projeto de um supermercado, por exemplo, exige que se distribua, da melhor forma possível, uma série de objetos idênticos — prateleiras, refrigeradores etc. — em uma certa área. Esse tipo de sistema, denominado Projeto Assistido pelo Computador (PAC), pode se apresentar com vários níveis de sofisticação. Mas, qualquer que seja seu grau de complexidade, o PAC baseia-se em duas operações básicas: traçar retas e remodelar figuras prefixadas.

Este artigo irá ajudá-lo a construir duas versões simples de um PAC.

TRAÇANDO RETAS

A técnica que examinaremos agora reproduz a maneira mais simples de se executar um desenho com o auxílio de um lápis e uma régua.

Primeiro, fazemos o traço inicial. Depois, a partir de uma das extremidades desse traço, desenhamos outra reta e assim por diante, sempre ligando a ponta do último segmento a um ponto imaginário. No computador, fixamos um ponto na tela e levamos o cursor até a posição onde um segundo ponto será fixado. Este irá determinar a reta que deve ser traçada.

Para facilitar ainda mais o processo, o computador traça uma reta a cada nova posição definida pelo cursor, apagando a anterior. Ao chegar à posição definitiva, o usuário digita algum tipo de código destinado a impedir que o computador apague a última linha, deixando-a fixa na tela. Em seguida, pode-se movimentar o cursor para outro ponto qualquer.

Experimente fazer desenhos com o programa que se segue.

```
10 BORDER 0: PAPER 0: INK 7:
CLS
15 LET inicio=80: LET estica=
130: LET desenha=210: LET fix
a=240
17 DRAW 0,175: DRAW 255,0:
DRAW 0,-175: DRAW -255,0
20 GOSUB inicio
40 GOSUB estica
50 IF INKEY$<>"q" THEN  GOTO
 40
60 STOP
90 OVER 1
100 LET x=128: LET y=86
115 PLOT x,y
120 RETURN
140 IF INKEY$=" " THEN  GOSUB
fixa: GOTO 150
145 GOSUB desenha
150 IF INKEY$="z" THEN  LET x=
x-1
160 IF INKEY$="x" THEN  LET x=
x+1
170 IF INKEY$="k" THEN  LET y=
y+1
180 IF INKEY$="m" THEN  LET y=
y-1
190 GOSUB desenha
200 RETURN
220 PLOT x,y: DRAW x-PEEK
23677,y-PEEK 23678
230 RETURN
250 OVER 0: GOSUB desenha
255 OVER 1
280 RETURN
```

```
10 PCLEAR 8:PMODE 4,5:PCLS:PMOD
E 4,1:PCLS:SCREEN 1,1
20 V=247:CX=127:CY=95:X=127:Y=9
5
30 GOSUB 100
40 FOR K=1 TO 4:PCOPY K+4 TO K:
NEXT
50 IF PEEK(338)<>191 THEN 30
60 CLS:END
100 IF PEEK(345)=V GOSUB 300
110 IF PEEK(341)=V AND Y>0 THEN
 Y=Y-1
120 IF PEEK(342)=V AND Y<191 TH
EN Y=Y+1
```

```
130 IF PEEK(343)=0 AND X>0 THEN
 X=X-1
140 IF PEEK(344)=0 AND X<255 TH
EN X=X+1
200 LINE(CX,CY)-(X,Y),PSET
210 RETURN
300 GOSUB 200
310 FOR K=1 TO 4:PCOPY K TO K+4
:NEXT
320 CX=X:CY=Y
330 RETURN
```

```
10  HGR2
20 CX = 140:CY = 96:X = CX:Y =
CY
100  GET A$
102  IF A$ = " " THEN CX = X:CY
 = Y: GOTO 110
103  HCOLOR= 0: HPLOT CX,CY TO
X,Y
110  IF A$ = "Z" AND X > 1 THEN
 X = X - 2
120  IF A$ = "X" AND X < 278 TH
EN X = X + 2
130  IF A$ = "." AND Y < 190 TH
EN Y = Y + 2
140  IF A$ = ";" AND Y > 1 THEN
 Y = Y - 2
155  IF A$ = "S" THEN  STOP
160  HCOLOR= 3: HPLOT CX,CY TO
X,Y
170  GOTO 100
```

```
10  HGR2
20 CX = 140:CY = 96:X = CX:Y =
CY
100  GET A$
102  IF A$ = " " THEN CX = X:CY
 = Y: GOTO 110
103  HCOLOR= 0: HPLOT CX,CY TO
X,Y
110  IF A$ =  CHR$ (8) AND X >
1 THEN X = X - 2
120  IF A$ =  CHR$ (21) AND X <
 278 THEN X = X + 2
130  IF A$ =  CHR$ (113) AND Y
< 190 THEN Y = Y + 2
140  IF A$ =  CHR$ (112) AND Y
> 1 THEN Y = Y - 2
155  IF A$ = "S" THEN  STOP
160  HCOLOR= 3: HPLOT CX,CY TO
X,Y
170  GOTO 100
```

```
10 SCREEN 2
20 CX=128:CY=96:X=CX:Y=CY
100 A$=INKEY$:IF A$="" THEN 100
103 IF A$<>" " THEN LINE (CX,CY
)-(X,Y),4 ELSE CX=X:CY=Y
110 IF ASC(A$)=29 AND X>2 THEN
X=X-2
120 IF ASC(A$)=28 AND X<255 THE
N X=X+2
130 IF ASC(A$)=31 AND Y<191 THE
N Y=Y+2
140 IF ASC(A$)=30 AND Y>1 THEN
Y=Y-2
160 LINE (CX,CY)-(X,Y),11
170 GOTO 100
```

O funcionamento do programa é bem simples. Ele fixa um ponto no centro da tela e outro em uma posição determinada por quatro teclas — **Z** (esquerda), **X** (direita), **K** (acima) e **M** (abaixo). O APPLE usa ; (acima), . (abaixo) e **S** (para interromper o programa). Os microcomputadores das linhas MSX e o TK-2000 utilizam as teclas de controle do cursor (flechinhas).

Uma reta é traçada continuamente entre os dois pontos assim determinados. Quando alcançar sua posição definitiva, ela será fixada na tela, bastando, para isso, que se pressione a barra de espaço. Você poderá, então, prosseguir em outra direção, usando as mesmas quatro teclas.

Digite agora estas linhas adicionais para conseguir o efeito de tirar o lápis do papel.

```
 15 LET inicio=80: LET estica=
 130: LET desenha=210: LET fix
 a=240: LET apaga=210
 17 GOSUB apaga
144 IF INKEY$="d" THEN  GOSUB
apaga: GOTO 150
310 CLS
320 DRAW 0,175: DRAW 255,0:
DRAW 0,-175: DRAW -255,0
330 GOSUB inicio
340 RETURN
```

```
105 IF PEEK(339)=191 THEN PMODE
 4,5:PCLS:PMODE 4,1
301 A$=INKEY$
302 A$=INKEY$:IF A$="" THEN 302
304 IF A$="D" THEN 320
```

```
150  IF A$ = "D" THEN CX = X:CY
 = Y
```

```
105  IF A$ = "D" THEN CX = X:CY
 = Y
```

Agora, ao pressionar a tecla **D**, a última linha traçada será apagada. O ponto onde o cursor se encontra será definido como inicial.

MONTAGEM DE UM DESENHO

Alguns tipos de PAC apresentam em um menu várias figuras previamente definidas. Se o problema fosse a decoração de uma casa, por exemplo, o menu incluiria os desenhos de móveis, esquadrias e outros elementos, que seriam transportados para a tela e ajustados à vontade ao projeto do decorador. Para tais ajustes, é possível recorrer a ampliações, reduções e rotações.

O programa a seguir oferece, em seu menu, cinco figuras geométricas. Elas podem ser fixadas na tela e manipuladas de acordo com as instruções acrescentadas após a listagem.

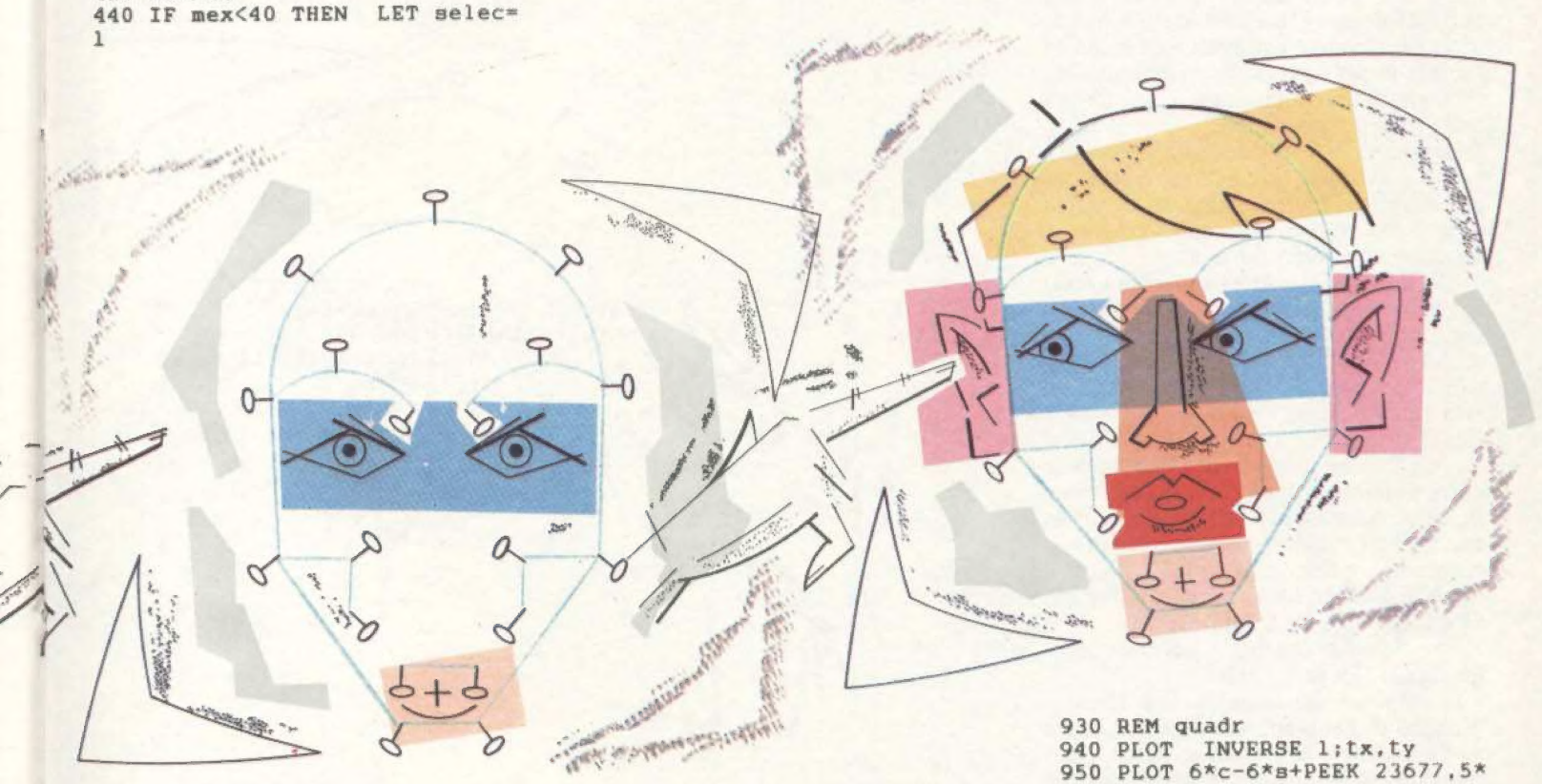
```
10 BORDER 0: PAPER 0: INK 7:
OVER 0: CLS
15 LET inicio=100: LET pick=
270: LET posicao=430: LET mod
ifica=490: LET seta=390: LET
desenha=640: LET fixa=700:
LET limpa=750: LET checa=820:
LET triang=860: LET quadr=930
: LET retang=1010: LET pent=
1090: LET hex=1180
17 GOSUB limpa
```

```
 40 GOSUB pick
 50 GOSUB posicao
 60 GOSUB modifica
 70 IF INKEY$<>"q" THEN  GOTO
  40
 80 STOP
110 OVER 0
120 LET x=30: LET y=50: LET ph
i=0: LET scal=1
130 LET mex=120: LET flag=0:
LET selec=0
140 LET c=scal*COS (phi)
150 LET s=scal*SIN (phi)
170 LET tx=32: LET ty=25:
GOSUB triang
190 LET tx=70: LET ty=25:
GOSUB quadr
210 LET tx=120: LET ty=25:
GOSUB retang
220 LET tx=180: LET ty=25:
GOSUB pent
230 LET tx=230: LET ty=25:
GOSUB hex
240 OVER 1
```

```
250 PRINT  OVER 1; INK 6;AT 20
,mex/8;"^"
260 RETURN
300 GOSUB seta
310 GOSUB checa
320 IF CODE INKEY$=8 THEN  LET
mex=mex-8
330 IF CODE INKEY$=9 THEN  LET
mex=mex+8
340 GOSUB seta
350 IF INKEY$="s" THEN  LET fl
ag=1
360 IF flag=0 THEN  GOTO 300
370 GOSUB seta
380 RETURN
400 PRINT  INK 6; OVER 1;AT 20
,mex/8;"^"
410 RETURN
440 IF mex<40 THEN  LET selec=
1
```

```
x+3
570 IF INKEY$="k" THEN  LET y=
y+3
580 IF INKEY$="m" THEN  LET y=
y-3
590 IF INKEY$="n" THEN  LET sc
al=scal*1.1
600 IF INKEY$="j" THEN  LET sc
al=scal/1.1
610 IF INKEY$="h" THEN  LET ph
i=phi+(6*PI/180)
620 IF INKEY$="b" THEN  LET ph
i=phi-(6*PI/180)
630 RETURN
645 LET tx=x: LET ty=y
650 IF selec=1 THEN  GOSUB tri
ang
```

```
770 DRAW 0,175: DRAW 255,0:
DRAW 0,-175: DRAW -255,0
780 GOSUB inicio
790 RETURN
820 REM checa
830 IF INKEY$="c" THEN  GOSUB
limpa
840 IF INKEY$="e" THEN  STOP
850 RETURN
860 REM triang
870 PLOT  INVERSE 1;tx,ty
880 PLOT -7*s+PEEK 23677,6*c+
PEEK 23678
890 DRAW -6*c+11*s,-5*s-9*c
900 DRAW 12*c,10*s
910 DRAW -6*c-11*s,-5*s+9*c
920 RETURN
```



```
450 IF mex>=40 AND mex<70 THEN
 LET selec=2
460 IF mex>=70 AND mex<150
THEN  LET selec=3
465 IF mex>=150 AND mex<180
THEN  LET selec=4
470 IF mex>=180 THEN  LET sele
c=5
480 RETURN
510 IF INKEY$="c" THEN  GOSUB
limpa: GOTO 517
515 GOSUB desenha
517 IF INKEY$=" " THEN  GOSUB
fixa: GOTO 520
518 GOSUB desenha
520 LET c=scal*COS (phi)
540 LET s=scal*SIN (phi)
550 IF INKEY$="z" THEN  LET x=
x-3
560 IF INKEY$="x" THEN  LET x=
```

```
660 IF selec=2 THEN  GOSUB qua
dr
670 IF selec=3 THEN  GOSUB ret
ang
675 IF selec=4 THEN  GOSUB pen
t
680 IF selec=5 THEN  GOSUB hex
685 FOR n=1 TO 30: NEXT n
690 RETURN
700 REM fixa
710 FOR n=1 TO 100: NEXT n
714 IF INKEY$="d" THEN  OVER 1
715 IF INKEY$<>"d" THEN  OVER
0
717 GOSUB desenha
720 PRINT AT 20,1;"
                    ": OVER 1
730 GOSUB inicio
740 RETURN
760 CLS
```

```
930 REM quadr
940 PLOT  INVERSE 1;tx,ty
950 PLOT 6*c-6*s+PEEK 23677,5*
s+5*c+PEEK 23678
960 DRAW -12*c,-10*s
970 DRAW 12*s,-10*c
980 DRAW 12*c,10*s
990 DRAW -12*s,10*c
1000 RETURN
1010 REM retang
1020 PLOT  INVERSE 1;tx,ty
1030 PLOT 12*c-6*s+PEEK 23677,1
0*s+5*c+PEEK 23678
1040 DRAW -24*c,-20*s
1050 DRAW 12*s,-10*c
1060 DRAW 24*c,20*s
1070 DRAW -12*s,10*c
1080 RETURN
1090 REM pent
1100 PLOT  INVERSE 1;tx,ty
1110 PLOT -10*s+PEEK 23677,9*c+
PEEK 23678
1120 DRAW -10*c+7*s,-8*s-6*c
1130 DRAW 4*c+13*s,3*s-11*c
1140 DRAW 12*c,10*s
```

```
1150 DRAW 4*c-13*s,3*s+11*c
1160 DRAW -10*c-7*s,-8*s+6*c
1170 RETURN
1180 REM hex
1190 PLOT  INVERSE 1;tx,ty
1200 PLOT -12*s+PEEK 23677,10*c
+PEEK 23678
1210 DRAW -10*c+6*s,-8*s-5*c
1220 DRAW 12*s,-10*c
1230 DRAW 10*c+6*s,8*s-5*c
1240 DRAW 10*c-6*s,8*s+5*c
1250 DRAW -12*s,10*c
1260 DRAW -10*c-6*s,-8*s+5*c
1270 RETURN
```

O programa funciona da maneira abaixo descrita:

A sub-rotina **início**, compreendida entre as linhas 110 e 260, define os valores de algumas variáveis e desenha as figuras-padrão na parte mais baixa da tela, chamando, para tanto, as sub-rotinas necessárias.

Da linha 300 à linha 380, o computador lê as teclas que movem a seta para a direita e para a esquerda e a tecla **S**, que faz com que a figura escolhida apareça no centro da tela.

As linhas 400 e 410 (sub-rotina **seta**) reimprimem a seta toda vez que uma tecla é pressionada.

As linhas 440 a 480 compõem a sub-rotina **posição**. Ela checa a posição da seta em relação às figuras e define a variável "figura".

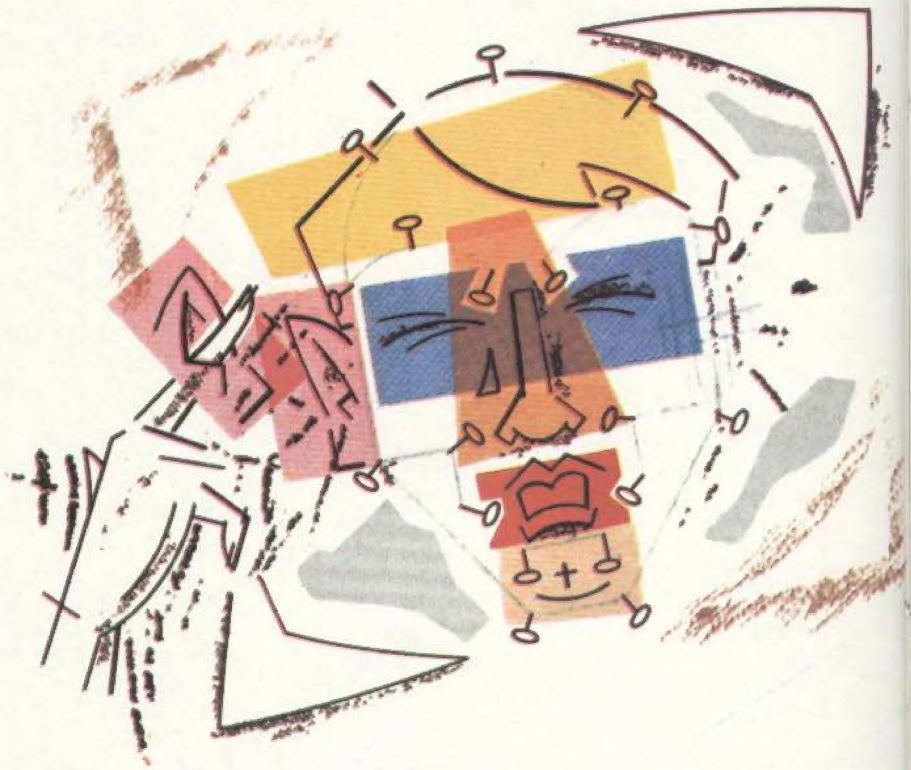
A sub-rotina **modifica** encontra-se entre as linhas 510 e 630. A figura escolhida é redesenhada de acordo com as teclas pressionadas. Ela pode ser movimentada usando-se as teclas **Z, X, K** e **M; N** e **J** ampliam e reduzem a figura; **H** e **B** promovem a rotação no sentido horário e anti-horário. A barra de espaço fixa a figura na tela.

As figuras são desenhadas pela combinação das sub-rotinas **desenha**, entre as linhas 645 e 690, com as sub-rotinas que contêm a fórmula de cada uma delas. A variável "figura" indica ao computador qual destas será usada.

A sub-rotina **fixa**, entre as linhas 710 e 740, fixa a figura na tela quando a tecla **D** é apertada. Nas linhas 760 a 790 está a sub-rotina **limpa**, que é chamada quando se pressiona **C**.

A sub-rotina **checagem** — linhas 820 a 840 — verifica se a tecla **E** foi acionada para interromper o programa.

```
10 PCLEAR 8:PMODE 4,5:PCLS:PMOD
E 4,1:PCLS:SCREEN 1,1
20 GOSUB 1000
30 GOSUB 2000
40 GOSUB 3000
50 GOSUB 4000
70 IF INKEY$<>"Q" THEN 40
80 CLS:END
1000 X=20:Y=131:PH=0:SC=1
1010 ME=122:ST=0:V1=247:V2=253
1020 C=SC:S=0:COLOR 5
1030 X=40:Y=177:GOSUB 5000
1040 X=80:Y=174:GOSUB 5100
1050 X=120:GOSUB 5200
1060 X=160:Y=170:GOSUB 5300
1070 X=200:GOSUB 5400
1080 FOR K=1 TO 4:PCOPY K TO K+
4:NEXT
1090 X=127:Y=85
1100 RETURN
2000 COLOR 0:GOSUB 2500
2010 A$=INKEY$:GOSUB 4500
2020 IF A$=CHR$(8) AND ME>32 TH
EN ME=ME-10
```



```
2030 IF A$=CHR$(9) AND ME<212 T
HEN ME=ME+10
2040 COLOR 5:GOSUB 2500
2050 IF A$<>"S" THEN 2000
2060 COLOR 5:GOSUB 2500
2070 RETURN
2500 DRAW"BM"+STR$(ME)+",190U5N
G2F2"
2510 RETURN
3000 SL=5:IF ME<192 THEN SL=4
3010 IF ME<152 THEN SL=3
3020 IF ME<102 THEN SL=2
3030 IF ME<62 THEN SL=1
3040 RETURN
4000 ON SL GOSUB 5000,5100,5200
,5300,5400
4010 IF PEEK(339)=191 THEN PCLS
:GOSUB 1000:RETURN
4020 IF PEEK(345)=V1 THEN FOR K
=1 TO 4:PCOPY K TO K+4:NEXT:RET
URN
4030 C=SC*COS(PH)
4040 S=SC*SIN(PH)
4050 IF PEEK(343)=V1 THEN X=X-1
4060 IF PEEK(344)=V1 THEN X=X+1
4070 IF PEEK(341)=V1 THEN Y=Y-1
4080 IF PEEK(342)=V1 THEN Y=Y+1
4090 IF PEEK(343)=V2 THEN SC=SC
*1.1
4100 IF PEEK(344)=V2 THEN SC=SC
/1.1
4110 IF PEEK(341)=V2 THEN PH=PH
+ATN(1)/7.5
4120 IF PEEK(342)=V2 THEN PH=PH
-ATN(1)/7.5
4130 FOR K=1 TO 4:PCOPYK+4 TO K
:NEXT
4150 IF PEEK(338)=V1 THEN RETUR
N ELSE 4000
4500 IF A$=CHR$(12) THEN PCLS:G
OSUB 1000
4510 IF A$="Q" THEN CLS:END
4520 RETURN
5000 LINE(X-7.2*S,Y-7.2*C)-(X-6
*C+3.6*S,Y+6*S+3.6*C),PSET
5010 LINE -(X+6*C+3.6*S,Y-6*S+3
.6*C),PSET
5020 LINE-(X-7.2*S,Y-7.2*C),PSE
T
5030 RETURN
5100 LINE(X+6*C-6*S,Y-6*S-6*C)-
(X-6*S-6*C,Y+6*S-6*C),PSET
5110 LINE -(X+6*S-6*C,Y+6*S+6*C
),PSET
5120 LINE -(X+6*S+6*C,Y+6*C-6*S
),PSET
5130 LINE -(X+6*C-6*S,Y-6*S-6*C
),PSET
5140 RETURN
5200 LINE (X+12*C-6*S,Y-6*C-12*
S)-(X-12*C-6*S,Y-6*C+12*S),PSET
5210 LINE -(X-12*C+6*S,Y+6*C+12
```

```
*S),PSET
5220 LINE -(X+12*C+6*S,Y+6*C-12
*S),PSET
5230 LINE -(X+12*C-6*S,Y-6*C-12
*S),PSET
5240 RETURN
5300 LINE (X-10.2*S,Y-10.2*C)-(
X-9.8*C-3.2*S,Y-3.2*C+9.8*S),PS
ET
5310 LINE -(X-6*C+10.2*S,Y+10.2
*C+6*S),PSET
5320 LINE -(X+6*C+10.2*S,Y+10.2
*C-6*S),PSET
5330 LINE -(X+9.8*C-3.2*S,Y-3.2
*C-9.8*S),PSET
5340 LINE -(X-10.2*S,Y-10.2*C),
PSET
5350 RETURN
5400 LINE(X-12*S,Y-12*C)-(X-10.
4*C-6*S,Y-6*C+10.4*S),PSET
5410 LINE -(X-10.4*C+6*S,Y+6*C+
10.4*S),PSET
5420 LINE -(X+12*S,Y+12*C),PSET
5430 LINE -(X+10.4*C+6*S,Y+6*C-
10.4*S),PSET
5440 LINE -(X+10.4*C-6*S,Y-6*C-
10.4*S),PSET
5450 LINE -(X-12*S,Y-12*C),PSET
5460 RETURN
```

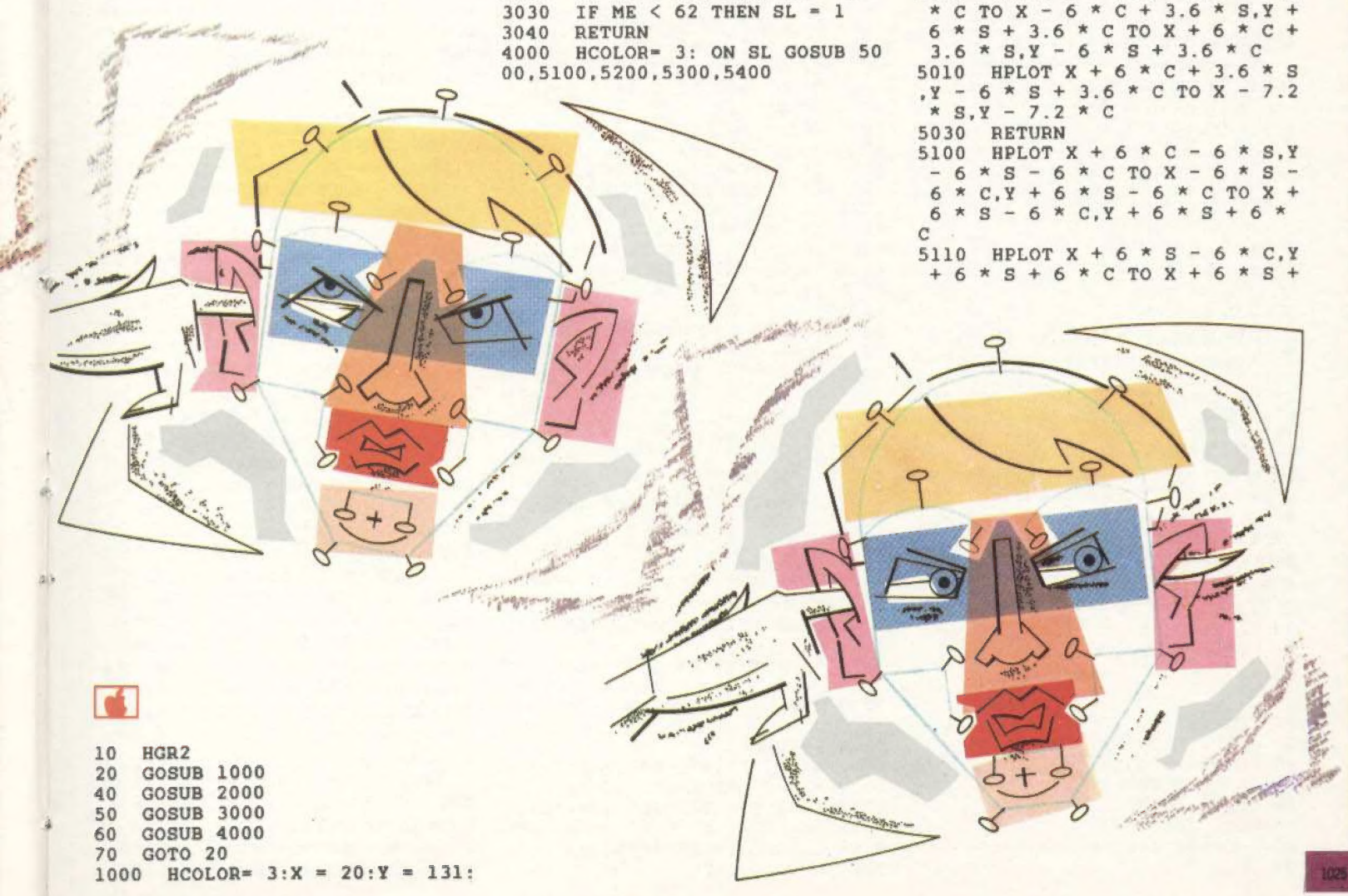
```
PH = 0:SC = 1
1010 ME = 122
1020 C = SC:S = 0
1030 X = 40:Y = 177: GOSUB 5000

1040 X = 80:Y = 174: GOSUB 5100

1050 X = 120: GOSUB 5200
1060 X = 160:Y = 170: GOSUB 530
0
1070 X = 200: GOSUB 5400
1090 X = 127:Y = 85
1100  RETURN
2000  HCOLOR= 3: GOSUB 2500
2005  GET A$
2010  HCOLOR= 0: GOSUB 2500
2020  IF A$ = "Z" AND ME > 32 T
HEN ME = ME - 10
2030  IF A$ = "X" AND ME < 212
THEN ME = ME + 10
2050  IF A$ <  > "S" THEN 2000
2070  RETURN
2500  HPLOT ME,190 TO ME,182 TO
 ME - 5,185: HPLOT ME,182 TO ME
 + 5,185
2510  RETURN
3000 SL = 5: IF ME < 192 THEN S
L = 4
3010  IF ME < 152 THEN SL = 3
3020  IF ME < 102 THEN SL = 2
3030  IF ME < 62 THEN SL = 1
3040  RETURN
4000  HCOLOR= 3: ON SL GOSUB 50
00,5100,5200,5300,5400
```

```
4010  GET A$
4015  HCOLOR= 0: ON SL GOSUB 50
00,5100,5200,5300,5400
4020  IF A$ = " " THEN  HCOLOR=
 3: ON SL GOSUB 5000,5100,5200,
5300,5400: RETURN
4030 C = SC *  COS (PH)
4040 S = SC *  SIN (PH)
4050  IF A$ = "Z" THEN X = X -
3
4060  IF A$ = "X" THEN X = X +
3
4070  IF A$ = ";" THEN Y = Y -
3
4080  IF A$ = "." THEN Y = Y +
3
4090  IF A$ = "M" THEN SC = SC
* 1.1
4100  IF A$ = "N" THEN SC = SC
/ 1.1
4110  IF A$ = "K" THEN PH = PH
+  ATN (1) / 7.5
4120  IF A$ = "L" THEN PH = PH
-  ATN (1) / 7.5
4140  IF A$ = "C" THEN  GOTO 10

4145  IF A$ = "F" THEN  TEXT :
END
4150  GOTO 4000
5000  HPLOT X - 7.2 * S,Y - 7.2
 * C TO X - 6 * C + 3.6 * S,Y +
 6 * S + 3.6 * C TO X + 6 * C +
 3.6 * S,Y - 6 * S + 3.6 * C
5010  HPLOT X + 6 * C + 3.6 * S
,Y - 6 * S + 3.6 * C TO X - 7.2
 * S,Y - 7.2 * C
5030  RETURN
5100  HPLOT X + 6 * C - 6 * S,Y
 - 6 * S - 6 * C TO X - 6 * S -
 6 * C,Y + 6 * S - 6 * C TO X +
 6 * S - 6 * C,Y + 6 * S + 6 *
C
5110  HPLOT X + 6 * S - 6 * C,Y
 + 6 * S + 6 * C TO X + 6 * S +
```



```
10  HGR2
20  GOSUB 1000
40  GOSUB 2000
50  GOSUB 3000
60  GOSUB 4000
70  GOTO 20
1000  HCOLOR= 3:X = 20:Y = 131:
```

```
 6 * C,Y + 6 * C - 6 * S TO X +
 6 * C - 6 * S,Y - 6 * S - 6 *
C
5140  RETURN
5200  HPLOT X + 12 * C - 6 * S,
Y - 6 * C - 12 * S TO X - 12 *
C - 6 * S,Y - 6 * C + 12 * S TO
 X - 12 * C + 6 * S,Y + 6 * C +
 12 * S
5210  HPLOT X - 12 * C + 6 * S,
Y + 6 * C + 12 * S TO X + 12 *
C + 6 * S,Y + 6 * C - 12 * S TO
 X + 12 * C - 6 * S,Y - 6 * C -
 12 * S
5240  RETURN
5300  HPLOT X - 10.2 * S,Y - 10
.2 * C TO X - 9.8 * C - 3.2 * S
,Y - 3.2 * C + 9.8 * S TO X - 6
 * C + 10.2 * S,Y + 10.2 * C +
6 * S
5310  HPLOT X - 6 * C + 10.2 *
S,Y + 10.2 * C + 6 * S TO X + 6
 * C + 10.2 * S,Y + 10.2 * C -
6 * S TO X + 9.8 * C - 3.2 * S,
Y - 3.2 * C - 9.8 * S
5320  HPLOT X + 9.8 * C - 3.2 *
 S,Y - 3.2 * C - 9.8 * S TO X -
 10.2 * S,Y - 10.2 * C
5350  RETURN
5400  HPLOT X - 12 * S,Y - 12 *
 C TO X - 10.4 * C - 6 * S,Y -
6 * C + 10.4 * S TO X - 10.4 *
C + 6 * S,Y + 6 * C + 10.4 * S
TO X + 12 * S,Y + 12 * C
5410  HPLOT X + 12 * S,Y + 12 *
 C TO X + 10.4 * C + 6 * S,Y +
6 * C - 10.4 * S TO X + 10.4 *
C - 6 * S,Y - 6 * C - 10.4 * S
5420  HPLOT X + 10.4 * C - 6 *
S,Y - 6 * C - 10.4 * S TO X - 1
2 * S,Y - 12 * C
5460  RETURN
```

O programa do TK-2000 é igual ao do Apple, com estas modificações:

```
2020  IF A$ =  CHR$ (8) AND ME
> 32 THEN ME = ME - 10
2030  IF A$ =  CHR$ (21) AND ME
 < 212 THEN ME = ME + 10
4050  IF A$ =  CHR$ (8) THEN X
= X - 3
4060  IF A$ =  CHR$ (21) THEN X
 = X + 3
4070  IF A$ =  CHR$ (112) THEN
Y = Y - 3
4080  IF A$ =  CHR$ (113) THEN
Y = Y + 3
```

```
10 SCREEN 2
20 GOSUB 1000
40 GOSUB 2000
50 GOSUB 3000
60 GOSUB 4000
70 GOTO 20
1000 COLOR 14:X=20:Y=131:PH=0:S
C=1
1010 ME=122
1020 C=SC:S=0
1030 X=40:Y=177:GOSUB 5000
1040 X=80:Y=174:GOSUB 5100
1050 X=120:GOSUB 5200
1060 X=160:Y=170:GOSUB 5300
1070 X=200:GOSUB 5400
1090 X=127:Y=85
1100 RETURN
2000 COLOR 14:GOSUB 2500
2005 A$=INKEY$:IF A$="" THEN 20
05
2010 COLOR 4:GOSUB 2500
2020 IF A$=CHR$(29) AND ME>32 T
HEN ME=ME-10
2030 IF A$=CHR$(28) AND ME<212
THEN ME=ME+10
2050 IF A$<>"S" THEN 2000
2070 RETURN
2500 DRAW"BM"+STR$(ME)+",190U5N
G2F2"
2510 RETURN
3000 SL=5:IF ME<192 THEN SL=4
3010 IF ME<152 THEN SL=3
3020 IF ME<102 THEN SL=2
3030 IF ME<62 THEN SL=1
3040 RETURN
4000 COLOR 14:ON SL GOSUB 5000,
5100,5200,5300,5400
4010 A$=INKEY$:IF A$="" THEN 40
10
4015 COLOR 4:ON SL GOSUB 5000,5
100,5200,5300,5400
4020 IF A$=" " THEN COLOR 14:ON
 SL GOSUB 5000,5100,5200,5300,5
400:RETURN
4030 C=SC*COS(PH)
4040 S=SC*SIN(PH)
4050 IF A$=CHR$(29) THEN X=X-3
4060 IF A$=CHR$(28) THEN X=X+3
4070 IF A$=CHR$(30) THEN Y=Y-3
4080 IF A$=CHR$(31) THEN Y=Y+3
4090 IF A$="M" THEN SC=SC*1.1
4100 IF A$="N" THEN SC=SC/1.1
4110 IF A$="K" THEN PH=PH+ATN(1
)/7.5
4120 IF A$="L" THEN PH=PH-ATN(1
)/7.5
4140 IF A$="C" THEN CLS:GOTO 20
4145 IF A$="F" THEN COLOR 15:EN
D
4150 GOTO 4000
5000 LINE (X-7.2*S,Y-7.2*C)-(X-
6*C+3.6*S,Y+6*S+3.6*C)
5010 LINE -(X+6*C+3.6*S,Y-6*S+3
.6*C)
5020 LINE -(X-7.2*S,Y-7.2*C)
5030 RETURN
5100 LINE (X+6*C-6*S,Y-6*S-6*C)
-(X-6*S-6*C,Y+6*S-6*C)
5110 LINE -(X+6*S-6*C,Y+6*S+6*C
)
5120 LINE -(X+6*S+6*C,Y+6*C-6*S
)
5130 LINE -(X+6*C-6*S,Y-6*S-6*C
)
5140 RETURN
5200 LINE (X+12*C-6*S,Y-6*C-12*
S)-(X-12*C-6*S,Y-6*C+12*S)
5210 LINE -(X-12*C+6*S,Y+6*C+12
*S)
5220 LINE -(X+12*C+6*S,Y+6*C-12
*S)
5230 LINE -(X+12*C-6*S,Y-6*C-12
*S)
5240 RETURN
5300 LINE (X-10.2*S,Y-10.2*C)-(
X-9.8*C-3.2*S,Y-3.2*C+9.8*S)
5310 LINE -(X-6*C+10.2*S,Y+10.2
*C+6*S)
5320 LINE -(X+6*C+10.2*S,Y+10.2
*C-6*S)
5330 LINE -(X+9.8*C-3.2*S,Y-3.2
*C-9.8*S)
5340 LINE -(X-10.2*S,Y-10.2*C)
5350 RETURN
5400 LINE (X-12*S,Y-12*C)-(X-10
.4*C-6*S,Y-6*C+10.4*S)
5410 LINE -(X-10.4*C+6*S,Y+6*C+
10.4*S)
5420 LINE -(X+12*S,Y+12*C)
5430 LINE -(X+10.4*C+6*S,Y+6*C-
10.4*S)
5440 LINE -(X+10.4*C-6*S,Y-6*C-
10.4*S)
5450 LINE -(X-12*S,Y-12*C)
5460 RETURN
```

A inicialização encontra-se entre as linhas 1000 e 1100, onde uma série de variáveis são definidas. As linhas 2000 a 2070 compreendem a sub-rotina encarregada de selecionar a figura. A seta é movimentada para a direita e para a esquerda através das teclas de controle (o Apple usa as mesmas teclas do programa anterior).

As linhas 3000 a 3040 identificam a figura que está sendo apontada pela seta, de acordo com a variável **ME**. Ao pressionar a tecla **S**, a figura escolhida aparecerá no centro da tela.

A sub-rotina de desenho situa-se entre as linhas 4000 e 4150. A linha 4000 desvia o programa para a sub-rotina que contém as instruções para desenhar a figura escolhida. A linha 4020 fixa a figura na tela se a barra de espaço for acionada.

As linhas 4090 e 4100 verificam se as teclas **M** e **N** foram pressionadas, ampliando ou reduzindo o desenho. A opção final é a rotação — linhas 4110 e 4120. As teclas **K** e **L** controlam a direção (para a esquerda e para a direita) em que será executada.

Ao pressionar **C** (**<CLEAR>** no TRS-Color), a tela será apagada, ficando pronta para o desenho seguinte.

As sub-rotinas restantes contêm instruções para a construção das cinco figuras. As linhas 5000 a 5030 desenham o triângulo; as linhas 5100 a 5140 traçam o quadrado; as linhas 5200 a 5240, o retângulo; as linhas 5300 a 5350, o pentágono e as linhas 5400 a 5460, o hexágono.

Tome cuidado para que seu desenho não saia da tela, o que poderia acarretar a interrupção do programa e uma mensagem de erro. Para interromper o programa, pressione **F**.

EFEITOS SONOROS COMPLEXOS

- COMO OBTER O EFEITO DE UM "TIRO LASER"
- MOVIMENTOS DO CONE
- PAUSAS
- CONTROLE DA DURAÇÃO

Os usuários do Apple e do TK-2000 já aprenderam a utilizar rotinas em código de máquina para produzir sons em seu micro. Verão agora como criar efeitos sonoros mais complexos.

No artigo da página 712, explicamos como contornar as limitações dos micros das linhas Apple e TK-2000 para que produzam sons. Examinaremos aqui a criação de efeitos mais sofisticados.

Com a rotina que se segue obtemos um ruído que pode ser utilizado como "tiro laser" em um jogo de ação. O truque consiste em combinar um som que vai se tornando mais agudo com outro que fica cada vez mais grave.

```
10 ORG 800
20 LDA #$00
30 STA $FF
40 LDA #$00
50 STA $FE
60 LOOP LDA #$00
70 STA $C030
80 LDX $FF
90 PAUSEA NOP
100 NOP
110 NOP
120 NOP
130 DEX
140 BNE PAUSEA
150 INC $FF
160 LDA #$00
170 STA $C030
180 LDX $FF
190 PAUSEB NOP
200 NOP
210 NOP
220 NOP
230 DEX
240 BNE PAUSEB
250 DEC $FE
260 BEQ FIM
270 JMP LOOP
280 FIM RTS
290 END
```

Para o míni-Assembler do TK-2000, a listagem é a seguinte:

```
0320-   LDA   #$00
0322-   STA   $FF
0324-   LDA   #$00
0326-   STA   $FE
0328-   LDA   #$00
032A-   STA   $C030
032D-   LDX   $FF
032F-   NOP
0330-   NOP
0331-   NOP
0332-   NOP
0333-   DEX
0334-   BNE   $032F
0336-   INC   $FF
0338-   LDA   #$00
033A-   STA   $C030
033D-   LDX   $FF
033F-   NOP
0340-   NOP
0341-   NOP
0342-   NOP
0343-   DEX
0344-   BNE   $033F
0346-   DEC   $FE
0348-   BEQ   $034D
034A-   JMP   $0328
034D-   RTS
```

Depois de estabelecido o endereço inicial, as quatro primeiras instruções colocam o valor zero nas posições $FF e $FE, que servirão como contadores. O registro A é usado para isso, porque não há no chip 6502 uma instrução que coloque um certo valor diretamente em um endereço de memória.

EMISSÃO DE SOM

A seguir, o registro A é novamente carregado com zero e a instrução **STA $C030** produz um movimento no cone do alto-falante.

O valor contido na posição de memória $FF é, então, colocado no registro X e as seis linhas seguintes provocam uma pausa com duração proporcional àquele valor. Como no programa anterior, a linha 130 encarrega-se de subtrair uma unidade de X e, enquanto este contador não tiver sido reduzido a zero, o laço entre as linhas 90 e 140 vai sendo repetido.

Após essa pausa, a instrução **INC $FF** soma a unidade ao conteúdo da posição $FF, a fim de que a pausa seguinte seja maior.

As linhas 160 e 170 produzem novamente um movimento do alto-falante e, logo depois, as linhas 190 a 240 provocam nova pausa, agora com duração proporcional ao conteúdo de $FE.

Ao contrário do que aconteceu na linha 150, a instrução **DEC $FE** subtrai uma unidade do conteúdo da posição $FE, fazendo com que, na próxima vez, a pausa seja menor.

FIM DA ROTINA

A instrução **BEQ FIM** termina o programa quando o conteúdo de $FE for reduzido a zero. O final da rotina ocorre com o desvio para o rótulo FIM, que contém a instrução **RTS**. Enquanto houver um número maior que zero em $FE, a instrução **JMP LOOP** faz o laço entre as linhas 60 e 270 ser repetido.

PAUSEA E PAUSEB

Como podemos observar, o programa produz movimentos do alto-falante — e, portanto, emite som — em duas ocasiões: nas linhas 70 e 170. Entre esses dois eventos ocorrem duas pausas: uma que é rotulada **PAUSEA** e a outra, **PAUSEB. PAUSEA** é controlada por $FF e vai aumentando de tamanho; **PAUSEB**, por $FE e vai diminuindo de tamanho. Combinam-se, dessa maneira, dois sons, um com freqüência crescente e outro com freqüência decrescente.

Na realidade, o laço **PAUSEA** completa 256 voltas na primeira vez, dando uma volta a mais cada vez que é executado. Como o contador $FF só usa um byte, 256 é representado pelo número zero, e 0 + 1 = 1. Assim, **PAUSEA** dá uma volta na segunda execução e uma volta a mais a cada nova execução. Já **PAUSEB** começa com 256 voltas e vai dando uma volta a menos a cada execução.

CONTROLE DA DURAÇÃO

Para diminuir a duração do efeito sonoro, basta retirar alguns comandos **NOP**. Retire quantidades iguais de cada laço de pausa, pois, do contrário, o timbre do som será alterado. Quem estiver usando o míni-Assembler precisará recalcular os desvios.

AVALANCHE: O VÔO DAS GAIVOTAS

Enquanto Willie tenta escapar dos perigos que o cercam, gaivotas levantam vôo e pairam no céu, acrescentando à aventura um detalhe digno de um produto comercial.

Nenhuma cena à beira-mar estaria completa sem uma revoada de gaivotas. Os próprios efeitos sonoros do jogo — que adicionaremos mais tarde ao programa — indicam a chegada das aves, que voarão pelo céu como parte do cenário de fundo.

Você deve estar pensando que este é um detalhe desnecessário, já que não interfere no desenvolvimento da aventura. Mas, certamente, são detalhes desse tipo que diferenciam um programa amador de um comercial. O tempo dos jogos em que dois bloquinhos rebatiam uma bola de tênis terminou.

Infelizmente, na versão de *Avalanche* para o TRS-Color, não há gaivotas pois sua inclusão sobrecarregaria o jogo com tabelas adicionais.

O programa que se segue não só imprime gaivotas na tela, como também faz com que elas batam as asas. Para criar esse efeito, utilizamos apenas duas estruturas de animação.

```
10 REM org 58751
20 REM qul ld a,(57349)
30 REM inc a
40 REM res 3,a
50 REM ld (57349),a
60 REM ld bc,57192
70 REM cp 4
80 REM jr c,qpt
90 REM ld bc,57208
100 REM qpt ld a,46
110 REM ld hl,42
120 REM push bc
130 REM call print
140 REM inc hl
150 REM call print
160 REM pop bc
170 REM ld hl,68
180 REM call print
190 REM inc hl
200 REM call print
210 REM ret
220 REM org 58217
230 REM print *
```

PREPARANDO O VÔO

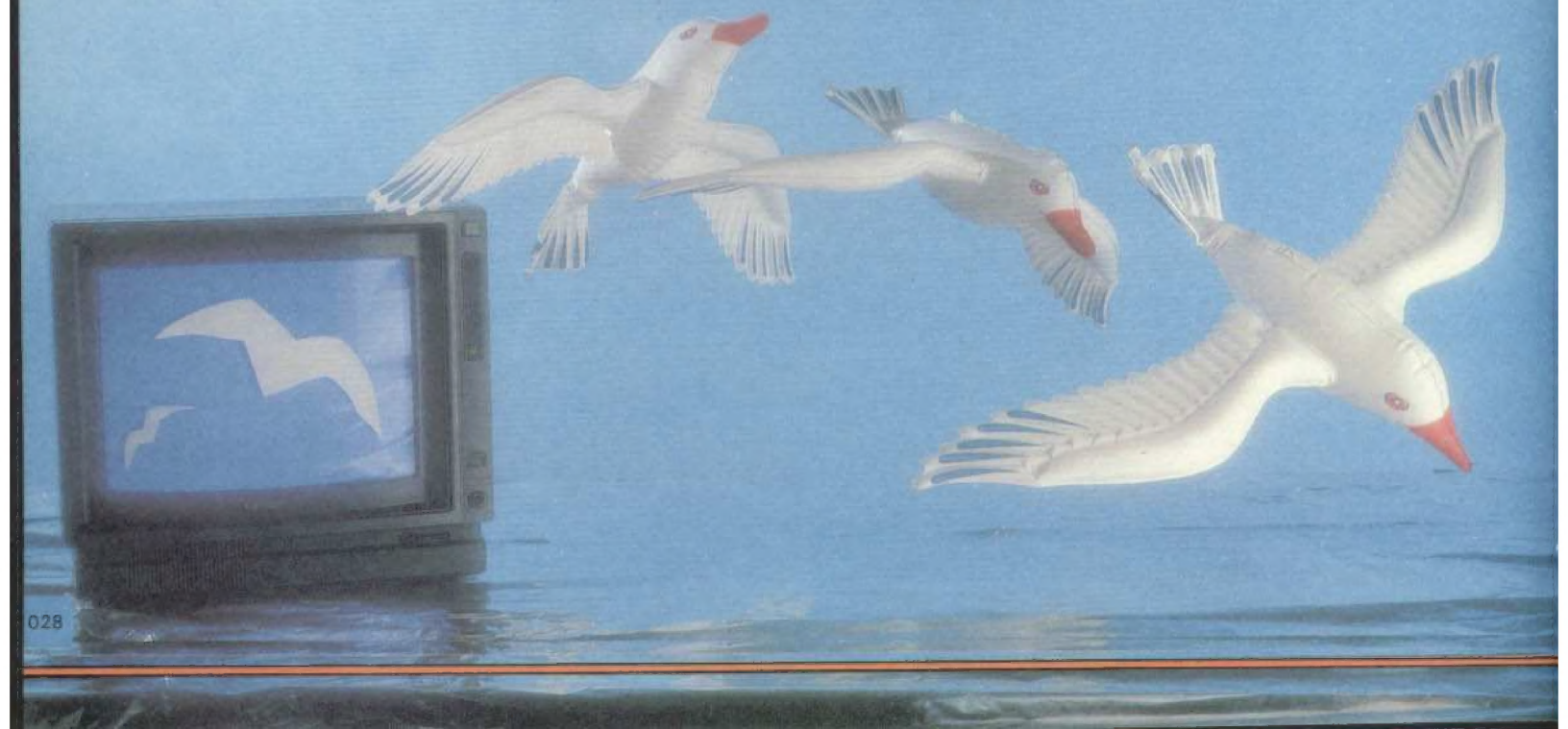
A posição de memória 57349 contém a chamada variável de atraso da gaivota. Essa variável impede que as gaivotas batam as asas muito rapidamente. O atraso completo é carregado e incrementado no acumulador A.

Para que as asas se movimentem para cima e para baixo adequadamente, o valor da variável de atraso deve oscilar num limite bem restrito. A instrução **res 3,a** ajusta o bit 3 do acumulador com o valor 0.

No momento em que a variável de atraso for incrementada com um valor superior a 7 — o bit 3 teria, então, o valor 1 —, o programa volta a ajustá-la com o valor 0. O resultado do incremento ou do ajuste para zero é novamente armazenado no endereço 57349.

Para fazer com que a ave pareça estar batendo as asas, colocamos na tabela de dados padrões de bits para duas gaivotas — uma com as asas para cima e outra com as asas para baixo. Os endereços iniciais dos dois conjuntos de dados são, respectivamente, 57192 e 57208. Cada gaivota compreende conjuntos de dezesseis bytes de dados.

O par de registros BC é carregado com o endereço do primeiro desses dois blocos de padrões ou dados. Em seguida, o conteúdo do acumulador — onde



VARIÁVEL DE ATRASO
COMO FAZER A GAIVOTA
BATER ASAS
DECOLAGEM
DEFINIÇÃO DO VÔO

o atraso da gaivota ainda está armazenado — é comparado com o número 4.

Uma instrução **cp** corresponde a uma subtração cujo resultado não foi armazenado. No nosso programa, o número 4 é subtraído do conteúdo do acumulador, mas não se armazena o resultado em nenhum lugar. Essa operação tem a função exclusiva de ajustar os indicadores ou balizas.

Em conseqüência, se o atraso da gaivota for menor do que 4 — ou seja, 0, 1, 2 ou 3 — o indicador denominado **carry** é ajustado com 1. Se o atraso da gaivota for maior do que 4, **carry** tem o valor 0.

A instrução **jr c** faz o processador saltar para o rótulo que a acompanha quando **carry** está ajustado com o valor 1. Assim, se o atraso da gaivota for menor do que 4, o processador pula a instrução seguinte e o endereço 57192 permanece no par de registros BC. Mas, se o atraso da gaivota for maior ou igual a 4 — e **carry** tiver, portanto, o valor 0 —, o salto não ocorre e o par de registros BC é carregado com o valor 57208, que corresponde ao endereço inicial dos padrões da segunda gaivota na tabela de dados.

DECOLAGEM

O acumulador A é carregado com 46, para que a cor da gaivota seja ajustada, e o par HL recebe a posição na tela da primeira gaivota, 42.

O conteúdo do par BC é então guardado na pilha. Para complicar um pouco mais, temos duas gaivotas tanto na tela como na tabela de dados. A tabela contém os padrões de uma gaivota com as asas para cima e outra com as asas para baixo, enquanto a tela exibe a mesma gaivota, impressa duas vezes em dois lugares diferentes. Como se utiliza uma só imagem, as gaivotas batem asas em sincronia.

Por esse motivo, o apontador de dados é preservado na pilha, enquanto a primeira gaivota está sendo impressa na tela. A rotina **print** incrementa esse apontador automaticamente durante a impressão dos dezesseis bytes de dados que são necessários para completar o desenho de uma gaivota. Ao ser chamada, a rotina **print** coloca na tela oito bytes de dados seguindo a posição inicial definida pelo par de registros BC. O conteúdo de BC é incrementado a cada byte impresso.

Depois da impressão da primeira parte da gaivota, o par HL é incrementado para apontar a posição na tela imediatamente à direita. Em seguida, a rotina **print** é chamada outra vez.

Quando os dois blocos tiverem sido impressos, a primeira gaivota estará completa. O computador passa, então, à impressão da segunda. Para isso, o apontador de dados é recuperado da pilha e o par HL é ajustado para apontar a nova posição de impressão. Feitos esses ajustes, a rotina **print** é chamada outra vez. Todo o processo já descrito se repete, até que se complete o desenho da segunda gaivota.

A listagem que você digitará a seguir não faz parte do programa principal. Este é composto por quatro rotinas, três das quais já publicadas. Apresentaremos a última delas no artigo que encerra a série *Avalanche*.

Este programa e todos os que o seguem, a partir do presente artigo, constituem sub-rotinas isoladas que irão ser chamadas pela quarta rotina do programa principal. Como esta será montada na seqüência da terceira rotina, a listagem que fornecemos aqui começa no endereço −11380. Deixamos, assim, um espaço reservado para a última parte do programa principal do jogo.

Esta rotina, além de colocar as gaivotas na tela, faz com que elas batam as asas para cima e para baixo usando só duas estruturas de animação.

```
10    org 54156
20    qui ld a,(-5206)
30    inc a
40    res 3,a
50    ld (-5206),a
60    ld b,28
70    cp 4
80    jr c,gpt
90    ld b,32
100    gpt push bc
110    ld a,b
120    ld hl,(62407)
130    ld de,42
140    add hl,de
150    push hl
160    push af
170    call 77
180    pop af
190    pop hl
200    inc hl
210    add a,2
220    call 77
230    pop bc
240    ld a,b
250    ld hl,(62407)
260    ld de,68
270    add hl,de
280    push hl
290    push af
300    call 77
310    pop af
320    pop hl
330    inc hl
340    add a,2
350    call 77
360    ret
370    end
```

DEFINIÇÃO DO VÔO

A posição de memória −5206 contém a variável de atraso da gaivota, que tem a função de impedir que as aves batam as asas muito rapidamente. O atraso completo é carregado e incrementado no acumulador A.

Para que as asas se movimentem para cima e para baixo de maneira adequada, o atraso da gaivota deve oscilar numa faixa restrita. A instrução **res 3,a** se encarrega disso, ajustando o bit 3 do acumulador com o valor 0. Analisando essa instrução, verifica-se que a variável de atraso será reajustada ao valor 0 sempre que ela for incrementada com um valor superior a 7. Nesse caso, o resultado do incremento e do ajuste para zero é armazenado novamente no endereço −5206.

Para dar a impressão de que a gaivota está batendo as asas, utilizamos padrões de bits para duas gaivotas — uma com as asas para cima e outra com as asas para baixo.

Os códigos dos padrões para a primeira gaivota são 28 e 30, e para a segunda, 32 e 34. Cada padrão compreende oito bytes de dados e cada gaivota possui dois padrões.

O registro B é carregado com o código do primeiro padrão da primeira gaivota. Depois, o conteúdo do acumulador — onde ainda está o atraso da gaivota — é comparado com o número 4.

Uma instrução **cp** equivale a uma subtração cujo resultado não foi armazenado. O número 4 é subtraído do conteúdo do acumulador, mas não se armazena o resultado em nenhum lugar. Essa operação tem a função exclusiva de ajustar os indicadores ou balizas (**flags**). Se o atraso da gaivota for menor do que 4, o indicador chamado **carry** é ajustado com o valor 1; se for maior ou igual a 4, **carry** não é ajustado.

A instrução **jr, c,gpt** faz o processador saltar para o rótulo **gpt** se **carry** foi

ajustado com 1. Assim, se o atraso da gaivota for menor do que 4, o processador salta a instrução seguinte e o código 28 permanece em B.

Se o atraso da gaivota for maior ou igual a 4, **carry** não foi ajustado e o salto não ocorre. O registro B é, então, carregado com o código 32, correspondente ao primeiro padrão da segunda gaivota na tabela de padrões.

DECOLAGEM

A rotina **gpt** — incumbida de colocar as duas gaivotas na tela — começa armazenando na pilha o código do primeiro padrão da gaivota que está sendo impressa. Isso é feito porque, como temos que imprimir a gaivota duas vezes, esse código sofrerá alteração durante a primeira impressão.

Para imprimir uma gaivota na tela, recorremos à rotina 77 da ROM, já utilizada em partes anteriores do jogo. Antes, precisamos ajustar seus parâmetros. O acumulador deve conter o código do padrão e o par de registros HL, o endereço na tabela de nomes onde vamos colocá-lo — ou seja, a posição de impressão na tela.

O código do primeiro padrão da gaivota é transferido do registro B para o acumulador. Em seguida, o par HL é carregado com o endereço inicial da tabela de nomes, que está armazenado nos endereços 62407 e 62408 da RAM. Adicionando-se o número 42 a esse endereço, por meio do par de registros DE, obtém-se a posição na tela da metade inicial da primeira gaivota.

Os conteúdos do par de registros HL e do acumulador são preservados na pilha, uma vez que podem ser alterados pela rotina 77. Esta é chamada logo em seguida e imprime a metade esquerda da primeira gaivota.

O apontador da posição na tela é recuperado da pilha, voltando ao par HL, enquanto o código do padrão retorna ao acumulador. Em seguida, o apontador em HL é incrementado para indicar a posição de impressão da metade direita. O número 2 é então adicionado ao acumulador, que passa a conter o código do padrão da segunda metade. Quando a primeira gaivota já está na tela, a rotina 77 é chamada.

O código do padrão da metade esquerda da gaivota é recuperado da pilha para o registro B, uma vez que essa mesma figura será novamente impressa em outra posição.

O procedimento é análogo ao já descrito, só que a adição do número 68 ao endereço inicial da tabela de nomes em HL indica a posição de impressão da segunda gaivota. A rotina 77 é chamada duas vezes, uma para cada metade, e o endereço da segunda gaivota se completa na tela também.

TESTANDO

Para testar a listagem aqui apresentada, espere a publicação da última rotina da série *Avalanche*. Se quiser se antecipar, digite uma pequena rotina em código de máquina que chame a rotina deste artigo várias vezes, com um pequeno atraso entre cada chamada. Tal atraso pode ser provocado por um laço qualquer. Depois disso, carregue a tabela de padrões e veja as gaivotas batendo suas asas.

SONS E RUÍDOS NO TRS-80

O TRS-80 não conta com comandos para a produção de efeitos sonoros. Dois pequenos programas em código de máquina, porém, permitem a criação de diversos sons nesse micro.

Muitos tipos de jogos e outros programas se tornarão bem mais interessantes e "profissionais" se você adicionar a eles música e efeitos sonoros, como ruídos de tiros, explosões, disparos de canhões de laser e coisas do gênero. Na verdade, esta é uma das razões pelas quais a maioria dos microcomputadores mais modernos tem comandos especiais em BASIC para o controle de sintetizadores internos de som, como é o caso do TRS-Color, do Spectrum, do MSX e do TK-2000.

Os micros compatíveis com a linha TRS-80 não contam com esse recurso, tão simples de usar. Mas isso não quer dizer que seja impossível programar efeitos sonoros e musicais nessas máquinas. Conhecendo alguns truques em linguagem Assembler, é fácil escrever pequenas rotinas para adicionar som aos seus programas.

FAÇA UM SOM

Fazer o programa não é tudo, porém. Normalmente, os microcomputadores da linha TRS-80 não têm alto-falante interno, como ocorre com o Apple, e nem possuem linha de conexão para o alto-falante da TV ou do monitor, como é o caso do MSX e do TK-2000.

Por essa razão, se quisermos produzir sons nesse micro, devemos utilizar a saída para o gravador cassete. Esse recurso vale também para os modelos da linha TRS-80 que dispõem de alto-falante interno (por exemplo, o Prológica CP-500), ligado em paralelo com a saída para cassete.

PORTA DE SAÍDA

Como é possível produzir efeitos sonoros numa saída desse tipo?

Um gravador de áudio é capaz de registrar apenas sons na faixa audível (entre 200 e 10.000 Hz, com os valores máximo e mínimo dependendo da qualidade do aparelho). Ora, uma interface especial na saída para o gravador transforma os impulsos binários (0 e 1), que constituem a "linguagem" interna do computador, em ondas elétricas com freqüência compatível. Em outras palavras, se você colocar uma fita gravada com um programa de computador para escutar, ouvirá uma "sinfonia" musical caótica, que corresponde aos sinais enviados pelo computador.

Essa interface de conversão digital-analógica está conectada internamente a uma *porta* do computador.

Uma porta é um canal de comunicação entre o microprocessador e o mundo externo. Em geral, uma porta faz a comunicação nos dois sentidos (entrada e saída), mas muitos micros usam determinadas portas só para um tipo de operação. É o caso da porta para gravador cassete, que é apenas de saída.

Em conseqüência, tudo o que necessitamos fazer para ouvir sons produzidos pela porta do gravador cassete é conectar o plugue do fio que normalmente é utilizado para gravar programas e dados (e que é chamado de AUX ou MIC) a um sistema de som, dotado de amplificador e alto-falante.

Quem tem um microcomputador da linha TRS-80, ou compatível, já dotado de alto-falante interno, não precisa nem mesmo tomar essas medidas.

SONS POR SOFTWARE

Para acionar uma porta de saída, usamos a instrução **OUT**, em linguagem Assembler, que se encarrega de enviar um byte para essa porta.

Para gerar sons com a instrução **OUT**, recorremos à velocidade do código de máquina. Tudo o que é preciso fazer, neste caso, é movimentar alternadamente o cone do alto-falante para fora e para dentro. Se fizermos isso uma vez, produziremos um clique do tipo que costuma ocorrer quando ligamos um aparelho de som. O truque consiste em repetir a operação seguidas vezes, e de modo bastante rápido. Se os movimentos forem suficientemente velozes, a sucessão de cliques vai parecer um zumbido. Quanto mais acelerado for o movimento de vaivém do cone, mais agudo será o som obtido.

Para escrever um programa destinado a criar qualquer tipo de efeito sonoro, é necessário que se conheçam alguns fatos básicos sobre a porta do gravador do micro TRS-80:

- A porta de saída do gravador tem o endereço 255 (FF em hexadecimal).
- Apenas o primeiro e o segundo bits do byte enviado à porta de saída são usados pela interface de conversão. O bit zero determina a saída de um impulso de som, ao passo que o bit um controla o motor do gravador.
- Quando o bit zero da porta é igualado a 1, ocorre um impulso audível positivo na linha AUX ou MIC do gravador. Quando o bit zero é igualado a 0, ocorre um impulso negativo.

Portanto, para conseguir uma aproximação de uma onda sinusoidal (tom puro de uma certa freqüência) na porta de saída, o software deve fazer o seguinte: um impulso positivo é enviado, e um pequeno período de espera é introduzido. Em seguida, um impulso negativo é enviado, e novo período de espera deve ocorrer.

Ao ser aumentado o período de espera, a freqüência do som gerado sofrerá uma diminuição; se, ao contrário, esse período for reduzido, a freqüência obtida será incrementada.

Para produzir um ruído ou qualquer outro efeito sonoro diferente, basta variar os dois intervalos de espera segundo algum padrão (ou, então, aleatoriamente: com isso conseguiremos o chamado "ruído branco", semelhante ao chiado de um rádio).

TONS PUROS

Apresentamos a seguir uma sub-rotina em linguagem de máquina que possibilita a produção de tons puros (bipes), com freqüência e duração individualmente controláveis.

Para facilitar o uso dessa rotina por programas em BASIC, a sub-rotina é inteiramente realocável, e é colocada em uma parte protegida da memória por uma rotina de inicialização, em BASIC, que precisa ser chamada apenas uma vez. Uma segunda rotina, também em BASIC, invoca a sub-rotina em código de máquina, por meio do comando **USR**.

É possível armazenar a rotina em có-

- PRODUZA SONS NO TRS-80
- INTERFACE DE CONVERSÃO
- A PORTA DE SAÍDA
- A INSTRUÇÃO OUT
- TONS MUSICAIS
- ROTINA DE PRODUÇÃO DE SONS
- PROGRAMA DE TESTE
- COMO VARIAR A FREQÜÊNCIA
- TIROS E EXPLOSÕES

digo de máquina em uma parte protegida da memória. Para isso, reinicialize o computador e forneça o limite mínimo da memória protegida quando ele apresentar na tela a questão "MEM USADA?" ou "MEMSIZE?". No exemplo a seguir, imaginamos que o computador tem 32Kbytes, e fixamos o limite mínimo da memória; assim, a rotina passará a ser armazenada na locação 49001. Para usar outro limite de memória, altere o valor da variável **IM** na linha 1010 do programa de inicialização. O valor sugerido para um computador com 48Kbytes de RAM seria 65000.

A versão aqui apresentada funciona em microcomputadores da linha TRS-80 com BASIC Nível II (para cassete):

```
1000 ' ROTINA DE INICIALIZACAO
1010 IM=49000
1020 FOR I=1 TO 35
1030 READ N:POKE IM+I,N
1040 NEXT N
1050 DATA 205,127,10,14,255,6
1060 DATA 1,237,65,17,29,0,27
1070 DATA 122,179,32,251,6,2
1080 DATA 237,65,17,29,0,27
1090 DATA 122,179,32,251,43
1100 DATA 124,181,32,227,201
1110 POKE 16527,INT(IM/256) :
POKE 16526,IM-INT(IM/256)*256+1
1120 RETURN
```

A rotina a ser chamada no momento de produção do som é assim:

```
2000 ' SUBROTINA PARA SOM
2010 CY=FQ*DR:N=29483/FQ
2020 POKE IM+11,N:POKE IM+23,N
2030 I=USR(CY):RETURN
```

Note que a rotina exige dois argumentos, que devem ser fornecidos pelo programa que a chamou: **DR** é a duração, em segundos, e **FQ**, a freqüência do som, em hertz (ciclos por segundo). Apresentamos a seguir um pequeno programa de teste, que exemplifica a maneira de usar as rotinas anteriores.

```
10 'PROGRAMA DE TESTE
20 GOSUB 1000 :'INICIALIZACAO
30 CLS:INPUT"DURACAO (S)";DR
40 INPUT"FREQUENCIA (HZ)";FQ
50 GOSUB 2000:GOTO 30
```

Para rodar o programa em micros com Disk BASIC (para disquete), faça as seguintes modificações:

```
1110 DEFUSR0=IM+1
2040 I=USR0(CY):RETURN
```

Os programas apresentados funcionam da seguinte maneira: a rotina de inicialização lê os códigos decimais correspondentes à rotina em linguagem de máquina, e os coloca na memória protegida por meio do comando **POKE**.

Antes de terminar, a rotina define o endereço de partida da rotina de máquina, que é, então, comunicado ao interpretador (par de apontadores 16526 e 16527, na versão destinada a gravador cassete, e DEFUSR, na versão adaptada para disco).

Na pequena rotina de produção de sons, inicialmente é calculado o número total de ciclos (positivo/negativo) necessário para produzir a duração pedida. Esse valor é armazenado na variável **CY**, que é passada à rotina **USR** por meio de seu argumento.

A duração total da pausa a ser executada entre os ciclos (**N**) é também calculada e fornecida à rotina diretamente, mediante dois comandos **POKE**, nos pontos onde é necessária.

A última linha da rotina em linguagem BASIC se encarrega de chamar a rotina em código de máquina, e retorna depois de ser executada.

TIROS, EXPLOSÕES E CHIADOS

O próximo programa funciona segundo o mesmo princípio que o anterior, só que o intervalo entre as fases positiva e negativa da onda sonora é alterado aleatoriamente a cada instante. Ao invés de um som puro (freqüência constante em toda a duração), esse tipo de intervalo produz um impulso de "ruído branco", que tem como característica uma mistura de todas as freqüências (daí seu nome, em analogia com a cor branca). Portanto, resta ao software controlar apenas a duração total do impulso sonoro. Durações muito curtas produzem um efeito sonoro semelhante a um tiro; durações um pouco maiores, algo semelhante a uma explosão, e durações muito longas, um chiado que lembra a estática de rádio (ou, se você quiser, o barulho produzido pelo reator de um avião a jato). Todos esses efeitos têm suas aplicações em programas de jogos.

FUNCIONAMENTO DA ROTINA

Da mesma maneira que a anterior, a sub-rotina é inteiramente realocável, e é colocada em uma parte protegida da memória por uma rotina de inicialização em BASIC, que precisa ser chamada apenas uma vez. Uma segunda rotina, também em BASIC, invoca a sub-rotina em código de máquina, por intermédio do comando **USR**.

A versão a seguir destina-se a microcomputadores da linha TRS-80 com BASIC Nível II (para cassete). A rotina de inicialização é assim:

```
1000 ' ROTINA DE INICIALIZACAO
1010 IR=49000
1020 FOR I=1 TO 26
1030 READ N:POKE IR+I,N
1040 NEXT N
1050 DATA 205,127,10,62,1,211
1060 DATA 255,237,95,87,71,16
1070 DATA 254,62,2,211,255,66
1080 DATA 16,254,43,124,181,32
1090 DATA 234,201
1110 POKE 16527,INT(IR/256) :
POKE 16526,IR-INT(IR/256)*256+1
1120 RETURN
```

A rotina de produção de sons:

```
2000 ' SUBROTINA PARA RUIDO
2010 N=DR*1000
2020 I=USR(N):RETURN
```

E um pequeno programa de teste:

```
10 'PROGRAMA DE TESTE
20 GOSUB 1000 :'INICIALIZACAO
30 CLS:INPUT"DURACAO (S)";DR
40 GOSUB 2000:GOTO 30
```

Finalmente, para rodar o programa em micros com Disk BASIC (para disquete), faça as seguintes modificações:

```
1110 DEFUSR0=IR+1
2040 I=USR0(N):RETURN
```

Se você quiser usar as duas rotinas em linguagem de máquina no mesmo programa, altere os endereços de partida IM e IR, de modo a evitar superposição (por exemplo, se IM = 49000, faça IR = 49036, no mínimo).

JOGOS DE GUERRA: O MAPA DA BATALHA

Chegou a hora de desenhar o mapa do campo de batalha. Vamos preenchê-lo com vilas, florestas e montanhas, para, depois, colocar as forças inimigas em suas posições iniciais.

No primeiro artigo desta série sobre jogos de guerra, criamos os blocos gráficos que irão representar as unidades militares no campo de batalha. Esses símbolos serão colocados e movidos em um mapa, para que possamos acompanhar o desenrolar da disputa e planejar nossa estratégia.

O MAPA

Capa e Espada terá uma matriz para representar o mapa. Este permanecerá em exibição durante todo o jogo, ocupando a maior parte do vídeo.

Como o mapa está sempre na tela, poderíamos dispensar a matriz, pois a memória de vídeo conterá todas as informações a respeito do mapa. Convém, no entanto, manter esses dados organizados em uma matriz, mesmo às custas de grande quantidade de memória. Será muito mais fácil obter e modificar as informações na matriz do que na memória de vídeo.

A matriz terá tantos elementos quantas forem as posições do mapa. Os mapas dos micros das linhas MSX, TRS-Color e Spectrum têm trinta por dezesseis posições; os do Apple e do TK-2000, vinte por dezoito.

AS TROPAS

Para controlar a posição das tropas no mapa e verificar se há obstáculos em seu caminho quando as movimentamos, precisamos de uma segunda matriz. Esta não só conterá a posição de cada uma das unidades, mas, também, toda e qualquer informação referente a elas.

Em *Capa e Espada*, a matriz da tropa conterá ainda informações relacionadas aos combates, ao movimento etc. O conteúdo de uma matriz desse tipo depende da natureza do jogo. Você terá melhores informações sobre isso à medida que formos ampliando o programa.

DIMENSIONAMENTO DAS MATRIZES

As linhas seguintes dimensionam as matrizes do mapa e da tropa.

```
350 DIM m(16,30)
355 DIM T(16,9)
```

```
350 DIM M(16,30)
355 DIM T(15,8)
```

```
350  DIM M(18,20)
360  DIM T(16,9)
```

```
350 DIM M(16,30)
355 DIM T(16,9)
```

Os valores contidos na matriz do mapa asseguram que ele tenha o tamanho da tela. A matriz da tropa, por sua vez, é dimensionada de modo que possa conter nove tipos de informação a respeito de cada uma das dezesseis unidades que estão em combate.

COMO PREENCHER O MAPA

O próximo passo consiste em determinar os vários tipos de terreno que compõem a área mapeada, bem como a posição inicial de cada uma das unidades. Poderíamos ter optado por um mapa fixo — o que seria muito importante se quiséssemos reproduzir uma batalha famosa. Porém, é bem provável que os jogadores prefiram alterar o campo de batalha a cada jogo. Para fazê-lo sem complicações, utiliza-se o gerador de números aleatórios do micro.

A definição do terreno poderia resumir-se a uma simples escolha ao acaso entre os tipos possíveis. Porém, a distribuição de florestas e montanhas geralmente não é um simples fruto do acaso. Seria interessante que o programa pudesse reproduzir um padrão mais próximo do real, concentrando montanhas ou florestas em certas áreas.

Durante o planejamento do mapa de um jogo de guerra, é importante também considerar o tipo de terreno no qual esperamos que se dê a ação. Em *Capa e Espada*, por exemplo, pretende-se simular uma guerra medieval, com a maioria das batalhas sendo travadas em campo aberto. Nesse caso, não é interessante ter montanhas e florestas demais.

A ESCOLHA DO TERRENO

A rotina que listamos a seguir é essencialmente aleatória, embora haja um certo controle sobre a seleção, o que ajuda a dar uma aparência de maior realidade ao mapa.

```
 20 DEF FN r(x)=INT (RND*x)+1
800 REM Escolhe terreno
810 LET R=FN r(50)
820 IF R>5 THEN  LET R=0
830 IF R>4 THEN  LET R=3:
 RETURN
840 IF R>1 THEN  LET R=2
850 RETURN
```

```
15 R=RND(-TIME)
20 DEF FN R(X)=INT(RND(1)*X)+1
800 REM TERRENO
810 R=FN R(50)
820 IF R>5 THEN R=0
830 IF R>4 THEN R=3:RETURN
840 IF R>1 THEN R=2
850 RETURN
```

```
20  DEF  FN R(X) =  INT ( RND (
1) * X) + 1
800  REM  TERRENO
810 R =  FN R(50)
820  IF R > 5 THEN R = 0
830  IF R > 4 THEN R = 3: RETUR
N
840  IF R > 1 THEN R = 2
850  RETURN
```

```
800 REM   ESCOLHE O TERRENO
```

- MATRIZES DO MAPA
- MATRIZES DAS TROPAS
- COMO PREENCHER O MAPA
- POSICIONAMENTO DAS UNIDADES
- COMO COLOCAR OS BLOCOS GRÁFICOS NA TELA
- MOLDURA PARA O MAPA
- FATORES QUE AFETAM O MOVIMENTO DAS FORÇAS

```
810 R=RND(50)
820 IF R>5 THEN R=0
830 IF R>4 THEN R=3:RETURN
840 IF R>1 THEN R=2
850 RETURN
```

O programa utiliza apenas números aleatórios inteiros. Como o Apple, o TK-2000, o Spectrum e o MSX não têm uma função como a **RND**, a linha 20 **DEF**ine uma **F**unção para isto.

Em todos os programas, um número aleatório inteiro entre 1 e 50 é criado. A rotina de "escolha de terreno" seleciona um valor para a variável **R** de acordo com o número criado. Se esse número for maior que 5, **R** valerá 0, código de campo aberto. Se for 5, **R** será 3, código de montanha; se for 2, 3 ou 4, **R** será 2, valor que representa floresta; e, se for 1, **R** fica valendo 1, código de vila.

CONTROLANDO O ACASO

A rotina de "escolha de terreno" é chamada para definir cada posição da matriz do mapa ao longo de uma de suas dimensões. Como foi mencionado anteriormente, não é desejável que a distribuição seja totalmente aleatória. A rotina que se segue impõe um certo padrão à distribuição dos tipos de terreno, tornando o mapa mais real.

```
370 LET i$="NOSL"
470 REM   Cria
480 FOR i=1 TO 16: GOSUB 800:
LET m(i,1)=R: NEXT i
490 FOR i=1 TO 16
500 FOR j=2 TO 30
510 LET s=FN r(10)
520 IF s<8 THEN  GOSUB 800
525 IF s>=8 THEN  LET R=m(i,j-
1)
530 LET m(i,j)=R
540 IF R=3 AND j<30 THEN  LET
m(i,j+1)=4
550 IF m(i,j)<>0 AND m(i,j)<>3
THEN  PRINT AT i,j;CHR$ (m(i,j
)+143)
555 IF m(i,j)=3 AND j<>30 THEN
 PRINT AT i,j;CHR$ 146;AT i,j+
1;CHR$ 147
560 NEXT j
570 NEXT i
580 GOSUB 720
590 FOR i=1 TO 8
600 FOR j=1 TO 2: LET T(i,j)=2
: LET T(i+8,j)=2: NEXT j
610 FOR j=3 TO 4: READ T(i,j):
LET T(i+8,j)=T(i,j): NEXT j
620 READ mr
630 FOR j=0 TO 8 STEP 8
640 LET T(i+j,5)=mr+FN r(2)
650 LET T(i+j,6)=(FN r(100)*10
)+10
660 LET T(i+j,7)=T(i+j,6)
670 NEXT j
680 LET T(i,8)=15
690 LET T(i+8,8)=1
700 NEXT i
710 RETURN
```

```
370 I$="NWSE"
470 REM CRIAÇAO
480 FOR I=1 TO 16:GOSUB 800:M(I
,1)=R:NEXT I
490 FOR I=1 TO 16
500 FOR J=2 TO 30
510 S=FN R(10)
520 IF S<8 THEN GOSUB 800
525 IF S>=8 THEN R=M(I,J-1)
530 M(I,J)=R
540 IF R=3 AND J<30 THEN M(I,J+
1)=4
550 IF M(I,J)<>0 AND M(I,J)<>3
THEN LOCATE J,I:PRINT CHR$(M(I,
J)+223)
555 IF M(I,J)=3 AND J<> 30 THEN
 LOCATE J,I:PRINT CHR$(226)+CHR
$(227);
560 NEXT J,I
580 GOSUB 720
590 FOR I=1 TO 8
600 FOR J=1 TO 2:T(I,J)=2:T(I+8
,J)=2:NEXT J
610 FOR J=3 TO 4:READ T(I,J):T(
I+8,J)=T(I,J):NEXT J
620 READ MR
630 FOR J=0 TO 8 STEP 8
640 T(I+J,5)=MR+FN R(2)
650 T(I+J,6)=(FN R(100)*10)+10
660 T(I+J,7)=T(I+J,6)
670 NEXT J
680 T(I,8)=15
690 T(I+8,8)=1
700 NEXT I
710 RETURN
```

```
370 I$ = "NOSL"
470  REM  CRIACAO
480  FOR I = 1 TO 18: GOSUB 800
:M(I,1) = R: NEXT
490  FOR I = 1 TO 18
500  FOR J = 1 TO 20
510 S =  FN R(10)
520  IF S < 8 THEN  GOSUB 800
525  IF S >  = 8 THEN R = M(I,J
 - 1)
530 M(I,J) = R
540  IF R = 3 AND J < 20 THEN M
(I,J + 1) = 4
550  IF M(I,J) <  > 0 AND M(I,J
) <  > 3 THEN Y = I:X = J:N = M
(I,J) - 1: GOSUB 10000
560  IF M(I,J) = 3 AND J <  > 2
0 THEN Y = I:X = J:N = 2: GOSUB
 10000:X = J + 1:N = 3: GOSUB 1
0000
570  NEXT J,I
580  GOSUB 720
590  FOR I = 1 TO 8
```

```
600  FOR J = 1 TO 2:T(I,J) = 2:
T(I + 8,J) = 2: NEXT J
610  FOR J = 3 TO 4: READ T(I,J
):T(I + 8,J) = T(I,J): NEXT J
620  READ MR
630  FOR J = 0 TO 8 STEP 8
640 T(I + J,5) = MR +  FN R(2)
650 T(I + J,6) = ( FN R(100) *
10) + 10
660 T(I + J,7) = T(I + J,6)
670  NEXT J
680 T(I,8) = 18
690 T(I + 8,8) = 1
700  NEXT I
710  RETURN
10000 X = X * 2 - 2:N = N * 2:
FOR W = 0 TO 1:X = X + W:N = N
+ W: FOR V = 0 TO 7
10010  POKE T + (Y - 8 * (Y > 7
) - 8 * (Y > 15)) * 128 + 40 *
(Y > 7) + 40 * (Y > 15) + X + 1
024 * V, PEEK (EE + N * 8 + V)
10020  NEXT V,W: RETURN
```

```
370 I$="NOSL"
470 REM CRIAR
480 FOR I=1 TO 16:GOSUB 800:M(I
,1)=R:NEXT I
490 FOR I=1 TO 16
500 FOR J=2 TO 30
510 S=RND(10)
520 IF S<8 THEN GOSUB 800
525 IF S>=8 THEN R=M(I,J-1)
530 M(I,J)=R
540 IF R=3 AND J<30 THEN M(I,J+
1)=4
550 IF M(I,J)<>0 AND M(I,J)<>3
THEN LINE (J*8,I*8)-(J*8+7,I*8+
7),PRESET,BF:DRAW"BM"+STR$(J*8)
+","+STR$(I*8)+UC$(M(I,J))
555 IF M(I,J)=3 AND J<>30 THEN
DRAW"BM"+STR$(J*8)+","+STR$(I*8
)+UC$(3)+"BM"+STR$((J+1)*8)+","
+STR$(I*8)+UC$(4)
560 NEXT J,I
580 GOSUB 720
590 FOR I=1 TO 8
600 FOR J=1 TO 2:T(I,J)=2:T(I+8
,J)=2:NEXT J
610 FOR J=3 TO 4:READ T(I,J):T(
I+8,J)=T(I,J):NEXT J
620 READ MR
630 FOR J=0 TO 8 STEP 8
640 T(I+J,5)=MR+RND(2)
650 T(I+J,6)=(RND(100)*10)+10
660 T(I+J,7)=T(I+J,6)
670 NEXT J
680 T(I,8)=15
690 T(I+8,8)=1
700 NEXT I
710 RETURN
```

A rotina gera um novo número aleatório, S, para cada elemento restante da matriz. O valor de S, selecionado na linha 510, varia de 1 a 10. A linha 520 faz com que, em 70% das vezes, o novo bloco de terreno seja escolhido ao acaso — se S<8, o programa chama a sub-rotina de escolha de terreno. Os 30% restantes serão constituídos de blocos iguais aos vizinhos da esquerda. Esse procedimento tem como objetivo criar grupos de blocos no mapa. As linhas que vão de 550 a 570 colocam os blocos na tela.

POSICIONAMENTO DAS TROPAS

As posições dos combatentes são armazenadas nas matrizes das tropas como pares de coordenadas — horizontais e verticais.

Inicialmente, os adversários ficam em extremos opostos da tela. No campo de *Capa e Espada*, o jogador começa no extremo sul (margem inferior do mapa), e o computador, no norte (topo da tela).

Portanto, a coordenada vertical já está previamente determinada.

A coordenada horizontal precisa ser selecionada. Assim como para a definição do tipo de terreno, é interessante que haja um elemento de acaso nessa escolha. Contudo, algumas restrições devem ser feitas — precisamos impedir, por exemplo, que duas unidades ocupem o mesmo espaço.

A rotina que listamos a seguir seleciona a posição inicial de cada unidade, de ambos os lados. Em primeiro lugar, as tropas são selecionadas ao acaso. Depois, o mapa é dividido verticalmente em oito colunas. Cada coluna representa os limites dentro dos quais as unidades serão colocadas. A posição definitiva de todas elas é, então, selecionada de acordo com os limites da coluna.

```
 860 REM Dispoe tropas
 870 INK 2
 880 FOR m=1 TO 2
 890 LET s=1: LET r=1
 900 FOR k=1 TO 8
 910 REM  Loop
 920 LET s=FN r(8*m)
 930 IF T(s,9)<>0 THEN  GOTO
  910
 940 LET r=FN r(4)+r
 950 LET r=r-INT (r/30)
 960 LET T(s,9)=r
 970 INK m
 980 PRINT AT T(s,8),T(s,9);u$(
 s)
 990 NEXT k
1000 NEXT m
1010 RETURN
```

```
860 REM TROPAS
880 FOR M=1 TO 2
890 S=1:R=1
900 FOR K=1 TO 8
920 S=FN R(8*M)
930 IF T(S,9)<>0 THEN 920
940 R=FN R(4)+R
950 R=R-INT(R/30)
960 T(S,9)=R
980 LOCATE T(S,9),T(S,8):PRINT
CHR$(U(S))
990 NEXT K,M
1010 RETURN
```

```
860  REM  DISTRIBUICAO
880  FOR M = 1 TO 2
890 S = 1:R = 1
900  FOR K = 1 TO 8
920 S =  FN R(8 * M)
930  IF T(S,9) <  > 0 THEN 920
940 R =  FN R(3) + R
950 R = R -  INT (R / 20)
960 T(S,9) = R
980 Y = T(S,8):X = T(S,9):N =
VAL ( MID$ (U$,S - (M - 1) * 8,
1)) + (M - 1) * 5: GOSUB 10000
990  NEXT K,M
1010  RETURN
```

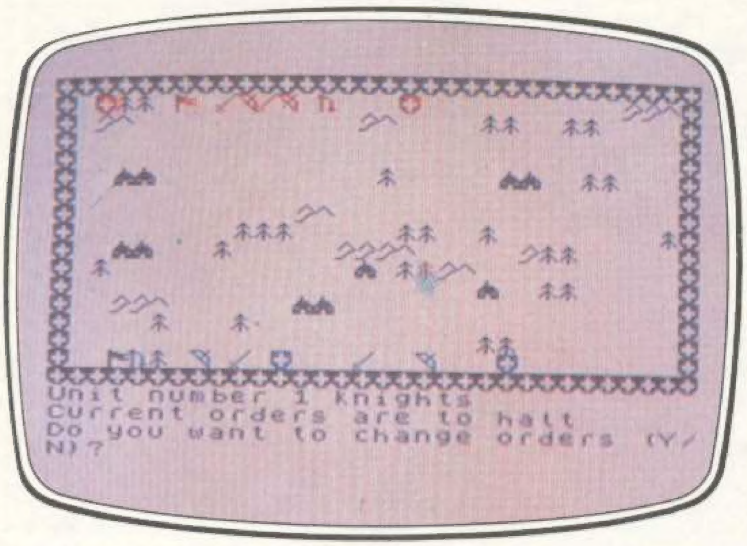
```
860 REM  DISPOE TROPAS
870 COLOR 2
880 FOR M=1 TO 2
890 S=1:R=1
900 FOR K=1 TO 8
910 REM
920 S=RND(8*M)
930 IF T(S,9)<>0 THEN 910
940 R=RND(4)+R
950 R=R-INT(R/30)
960 T(S,9)=R
970 COLOR M:IF M=1 THEN COLOR 3
980 DRAW"BM"+STR$(T(S,9)*8)+","
+STR$(T(S,8)*8):UU=VAL(MID$(U$,
S,1)):A$=UC$(UU):GOSUB 3000
990 NEXT K
1000 NEXT M
1010 RETURN
```

Como as oito unidades são escolhidas ao acaso, a posição inicial de cada uma varia e, assim, cada novo jogo é diferente do anterior.

Os dois exércitos estão armazenados em uma só matriz. Por isso, a mesma rotina pode ser usada para escolher a posição inicial de todas as tropas, bastando, apenas, um laço **FOR... NEXT** (linhas 880 e 1000). O laço também faz com que os dois exércitos apareçam em cores diferentes.

ÀS ARMAS

Uma vez determinadas as posições iniciais das unidades, elas devem ser colocadas no vídeo.



Capa e Espada: **mapa do campo de batalha no Spectrum.**

```
410 LET U$=CHR$ 148+CHR$ 149+
CHR$ 150+CHR$ 150+CHR$ 151+
CHR$ 151+CHR$ 152+CHR$ 152:
LET U$=U$+U$
```

```
410 U(1)=228:U(2)=229:U(3)=230:
U(4)=230:U(5)=231:U(6)=231:U(7)
=232:U(8)=232
```

```
410 U$ = "45667788"
```

```
410 U$="65778899":U$=U$+U$
```

Na linha 410 definimos um cordão **U$** para armazenar os códigos dos caracteres que representam cada unidade. No caso do MSX, utilizamos uma variável indexada **U( )**.

MOLDURA

A tela fica mais bonita — e menos confusa — se fizermos uma moldura para o campo de batalha.

```
720 REM   Borda Decorativa
730 FOR i=0 TO 16
740 PRINT AT i,0;CHR$ 150;AT i
,31;CHR$ 150
750 NEXT i
760 FOR i=0 TO 31
770 PRINT AT 0,i;CHR$ 150;AT
16,i;CHR$ 150
780 NEXT i
790 RETURN
```

```
720 REM BORDA
730 FOR I=0 TO 31
740 VPOKE BASE(5)+I,246
750 VPOKE BASE(5)+I+544,246
760 NEXT I:FOR I=0 TO 574 STEP
32
770 VPOKE BASE(5)+I,246
780 VPOKE BASE(5)+I-1,246
790 NEXT I:RETURN
```

```
720  REM  BORDA
730  FOR YY = 0 TO 19 STEP 19
740  FOR XX = 1 TO 20
750 N = 11:X = XX:Y = YY
760 X = X * 2 - 2:N = N * 2: FO
R W = 0 TO 1:X = X + W:N = N +
W: FOR V = 0 TO 7
770  POKE T + (Y - 8 * (Y > 7)
- 8 * (Y > 15)) * 128 + 40 * (Y
 > 7) + 40 * (Y > 15) + X + 102
4 * V, PEEK (E + N * 8 + V) - 1
28
780  NEXT V,W
790  NEXT XX,YY: RETURN
```

```
720 REM BORDA
730 LINE (0,128)-(255,135),PRES
ET,BF
740 LINE(248,0)-(255,191),PRESE
T,BF:LINE(4,4)-(252,132),PSET,B
790 RETURN
```

Essa rotina simplesmente desenha, repetidas vezes, em torno do mapa, o símbolo utilizado para os lanceiros, só que em outra cor.

MOBILIZAÇÃO DAS FORÇAS

Agora que o mapa está preparado, veremos como os jogadores movem suas unidades. Toda a mobilização decorre de ordens — explicadas no próximo artigo. Porém, antes que se possa completar o movimento, é preciso definir vários detalhes do jogo:

• Distância máxima, em número de posições, que cada unidade pode atingir em um só movimento. Em *Capa e Espada*, a mobilidade de uma unidade depende apenas do peso de sua armadura, mas, em outros jogos, fatores como moral, disciplina e cansaço, entre outros, podem ser levados em conta.

• Existência de vantagens e bônus. Em nosso programa, os bônus são dados apenas aos cavaleiros. Porém, é possível destiná-los também a unidades que realizem determinadas ações — como subir uma ladeira ou transportar carga — ou, ainda, que possuam um comandante muito habilidoso.

• Obstáculos ao movimento. O programador deve decidir que tipo de influência o terreno exerce sobre o movimento de tropas. Em nosso caso, todos os terrenos, menos o campo aberto, diminuem a distância máxima em uma unidade. Também é preciso verificar se há outra unidade no caminho.

• A borda do mapa foi alcançada?

Adicione mais esta rotina e o programa poderá movimentar as tropas levando em conta todos esses fatores.

```
1160 REM   Move unidade
1170 LET ox=T(b,8): LET oy=t(b,
9)
1175 LET z$=""
1180 IF m(T(b,8),T(b,9))<>0 THE
N  LET z$=CHR$ (143+m(T(b,8),T(
b,9)))
1190 LET D=5-T(b,4)
1200 IF b<3 OR b=9 OR b=10 THEN
  LET D=D+2
1210 LET v=T(b,2)-1
1215 LET up=0: LET al=v-2
```

```
1220 IF v/2-(INT (v/2))=0 THEN
 LET up=v-1: LET al=0
1230 REM  Repeticao
1240 LET nl=T(b,9)+al: LET np=T
(b,8)+up
1250 IF np<1 THEN  LET np=1
1260 IF np>15 THEN  LET np=15
1270 IF nl<1 THEN  LET nl=1
1280 IF nl>30 THEN  LET nl=30
1290 IF m(np,nl)>0 THEN  LET D=
D-1
1300 FOR k=1 TO 8
1310 IF (T(k,9)=nl AND T(k,8)=n
p AND k<>b) THEN  LET D=0
1315 IF (T(k+8,9)=nl AND T(k+8,
8)=np AND k+8<>b) THEN  LET D=0
1320 NEXT k
1330 IF d>0 THEN  LET T(b,9)=nl
: LET T(b,8)=np: LET D=D-1
1340 IF D<>0 THEN  GOTO 1230
1350 INK 0: PRINT AT ox,oy;z$
1360 INK cl: PRINT AT T(b,8),T(
b,9);u$(l)
1370 RETURN
```

```
1160 REM MOVE
1170 OX=T(B,8):OY=T(B,9)
1175 GH=32
1180 IF M(T(B,8),T(B,9)<>0 THEN
 GH=223+M(T(B,8),T(B,9))
1190 D=5-T(B,4)
1200 IF B<3 OR B=9 OR B=10 THEN
 D=D+2
1210 V=T(B,2)-1
1215 UP=0:AL=V-2
1220 IF V/2-(INT(V/2))=0 THEN U
P=V-1:AL=0
1240 N1=T(B,9)+AL:NP=T(B,8)+UP
1250 IF NP<1 THEN NP=1
1260 IF NP>15 THEN NP=15
1270 IF N1<1 THEN N1=1
1280 IF N1>30 THEN N1=30
1290 IF M(NP,N1)>0 THEN D=D-1
1300 FOR K=1 TO 8
1310 IF (T(K,9)=N1 AND T(K,8)=N
P AND K<>B) THEN D=0
1315 IF (T(K+8,9)=N1 AND T(K+8,
8)=NP AND K+8<>B) THEN D=0
1320 NEXT K
1330 IF D>0 THEN T(B,9)=N1:T(B,
8)=NP:D=D-1
1340 IF D<>0 THEN 1240
1350 LOCATE OY,OX:PRINT CHR$(GH
);
1360 LOCATE T(B,9),T(B,8):PRINT
 CHR$(U(I)+CL)
1370 RETURN
```

```
1160  REM  MOVE
1170 OX = T(B,8):OY = T(B,9)
1175 GH = 14
1180  IF M(T(B,8),T(B,9)) <  >
0 THEN GH = M(T(B,8),T(B,9)) -
1
1190 D = 5 - T(B,4)
1200  IF B < 3 OR B = 9 OR B =
10 THEN D = D + 2
1210 V = T(B,2) - 1
1215 UP = 0:AL = V - 2
1220  IF V / 2 - ( INT (V / 2))
 = 0 THEN UP = V - 1:AL = 0
1240 N1 = T(B,9) + AL:NP = T(B,
8) + UP
1250  IF NP < 1 THEN NP = 1
1260  IF NP > 18 THEN NP = 18
1270  IF N1 < 0 THEN N1 = 1
1280  IF N1 > 19 THEN N1 = 20
1290  IF M(NP,N1) > 0 THEN D =
D - 1
1300  FOR K = 1 TO 8
1310  IF (T(K,9) = N1 AND T(K,8
) = NP AND K <  > B) THEN D = 0
1315  IF (T(K + 8,9) = N1 AND T
(K + 8,8) = NP AND K + 8 <  > B
) THEN D = 0
1320  NEXT K
1330  IF D > 0 THEN T(B,9) = N1
:T(B,8) = NP:D = D - 1
1340  IF D <  > 0 THEN 1240
1350 X = OY:Y = OX:N = GH: GOSU
B 10000
1360 X = T(B,9):Y = T(B,8):N =
 VAL ( MID$ (U$,I - (I > 8) * 8
,1)) + (I > 8) * 5: GOSUB 10000
1370  RETURN
```

```
1160 REM  MOVE UNIDADE
1170 OX=T(B,8):OY=T(B,9)
1175 ZZ=0
1180 IF M(T(B,8),T(B,9))<>0 THE
N ZZ=M(T(B,8),T(B,9))
1190 D=5-T(B,4)
1200 IF B<3 OR B=9 OR B=10 THEN
 D=D+2
1210 V=T(B,2)-1
1215 UP=0:AL=V-2
1220 IF (V/2)-INT(V/2)=0 THEN U
P=V-1:AL=0
1230 REM
1240 NL=T(B,9)+AL:NP=T(B,8)+UP
1250 IF NP<1 THEN NP=1
1260 IF NP>15 THEN NP=15
1270 IF NL<1 THEN NL=1
1280 IF N>30 THEN NL=30
1290 IF M(NP,NL)>0 THEN D=D-1
1300 FOR K=1 TO 8
1310 IF (T(K,9)=NL AND T(K,8)=N
P AND K<>B) THEN D=0
1315 IF (T(K+8,9)=NL AND T(K+8,
8)=NP AND K+8<>B) THEN D=0
1320 NEXT K
1330 IF D>0 THEN T(B,9)=NL:T(B,
8)=NP:D=D-1
1340 IF D<>0 THEN 1230
1350 X9=OY*8:Y9=OX*8:IF ZZ<>0 T
HEN COLOR 4:LINE(X9,Y9)-(X9+7,Y
9+7),PRESET,BF:DRAW"BM"+STR$(X9
)+","+STR$(Y9)+UC$(ZZ) ELSE LIN
E (X9,Y9)-(X9+7,Y9+7),PRESET,BF
1360 COLOR CL:DRAW"BM"+STR$(T(B
,9)*8)+","+STR$(T(B,8)*8):UU=VA
L(MID$(U$,I,1)):A$=UC$(UU):GOSU
B 3000
1370 RETURN
```

Os testes vão aumentar, reduzir ou impedir completamente a mobilidade das tropas. A rotina primeiro "lembrase" da posição anterior e do tipo de terreno que havia naquela posição. Em seguida, calcula a direção e o deslocamento máximo.

As linhas 1230 a 1340 formam um laço que testa cada posição ao longo da trajetória da unidade, verificando se são ocupadas por tropas ou tipos de terreno que afetam o movimento. De acordo com o que esse laço encontrar, o deslocamento final é calculado. O laço se repete até que a distância ao ponto de destino seja zero.

Após decidir sobre o deslocamento, a rotina coloca o símbolo do terreno original na posição anterior e desenha a tropa na nova posição.

As listagens aqui apresentadas ainda não poderão ser completamente testadas. Como está, o programa apenas desenha o mapa, sendo interrompido por uma mensagem de "falta de dados". Com as rotinas do próximo artigo, que trata das ordens dadas pelo jogador, esse problema deixará de existir.

O que é uma simulação aleatória?

Os jogos de simulação geralmente envolvem uma metodologia de cálculo específica para a área que está sendo simulada. Na maioria das simulações isso abrange tanto fórmulas matemáticas quanto regras de decisão (regras heurísticas).

Entretanto, o jogo de simulação fica muito previsível se não contiver algum elemento de probabilidade (ou seja, um sorteio ao acaso realizado para uma equação matemática para uma regra de decisão). As situações passam a se repetir exatamente da mesma maneira frente a um determinado conjunto de condições.

Por isso, é importante usar o gerador interno de números aleatórios do computador (funções **RAND**, **RND** etc.) para sortear ao acaso o caminho a ser seguido ou o valor numérico de algum dado de simulação.

São exemplos do que pode ser gerado em programas de simulação:

- O nome de uma nave espacial e o de seu comandante.
- O número e a direção das saídas de uma câmara secreta em um jogo de aventuras.
- A permanência do jogador na prisão, no Jogo Imobiliário.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NOVOS LANÇAMENTOS, NOVOS SUCESSOS.

JÁ NAS LIVRARIAS

A FRATERNIDADE DA PEDRA
David Morrell

Um grupo secreto, sob direção de um padre armado, passa a agir contra o terrorismo. Mas, será que violência se combate com mais violência? Eis o dilema de Drew, agente da Lei envolvido com fatos e figuras do mundo real, num livro surpreendente do criador de Rambo.

AVENTUREIROS E MILIONÁRIOS
Clark Howard

No romance do Texas, a saga da descoberta de um poço de petróleo e o drama de um casal que herda um lote de terra aparentemente sem valor e enfrenta com coragem os poderosos do lugar, até vencer. Uma vitória antes de tudo moral, numa história forte e envolvente, com realistas cenas de amor.

CAÇADA SEM FIM
Bryan Forbes

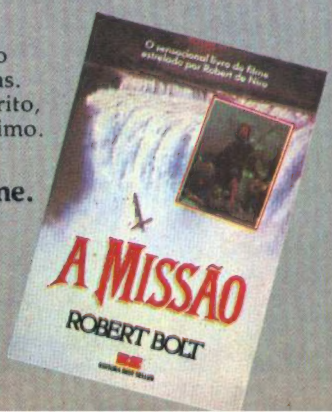
Uma brilhante história de espionagem envolvendo a KGB. Por que matar uma ex-espiã que já tinha sido desmascarada e torturada tempos atrás? Um agente inglês, seu antigo amante, enfrenta um desafio: descobrir por que ela foi morta... e por que *agora!*

PODER
Howard Fast

Um líder sindical com a volúpia do poder, a luta pelos direitos dos trabalhadores, nos Estados Unidos, e sua manipulação por corruptos e oportunistas; o jogo das ambições políticas. Admiravelmente escrito, um romance atualíssimo.

SUPERSEXO
Alexandra Penney

Se não for o primeiro, este vai ser o último e definitivo guia para o prazer que o leitor poderá seguir: um livro que derruba mitos, faz sugestões provocantes e propõe técnicas ousadas para se chegar ao supersexo, uma relação intensa e especial entre os casais, que não exclui o romantismo.



Não perca também: A MISSÃO, de Robert Bolt, o livro do filme.

EDITORA BEST SELLER